CS 104

AD 673010

A NUMERICAL INVESTIGATION OF THE SIMPLEX METHOD

BY

RICHARD H. BARTELS

TECHNICAL REPORT NO. CS 104
JULY 31, 1968

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

124

A Numerical Investigation of the Simplex Method*

By

Richard H. Bartels

Computer Science Department

Stanford University

*Research sponsored by the National Science Foundation under contract NSF GP 5962 and by the Office of Naval Research under contract N00014-67-A-0112-0029.

Acknowledgments

I am happy at having this chance to express my gratitude to:

[redacted] who, first as my fiancée and then as my wife, lifted my spirits through the final agonies of doctoral candidacy;

Gene Golub, my thesis advisor and my friend, who has given me measureless help, academically and privately, throughout my stay at Stanford;

Professor George E. Forsythe, whom I thank for his aid and advice to me during the course of my academic progress;

Professor George B. Dantzig, whose words of encouragement to me were very welcome;

And Miss Suzanne Stone, who under the pall of a deadline, did all of the typing which I required and maintained her composure throughout.

I would like to note that my work at the Computer Science Department has been financially supported by the National Science Foundation and the Office of Naval Research.

And I give my thanks to Doctor James H. Wilkinson and, again, to George B. Dantzig. They laid the foundations.

Table of Contents

1. Introduction and Summary of Results

Since 1947 linear programming problems have been growing ever more interesting to business, industry, government, and the military. Today few facets of the economy remain uninfluenced by decisions made with the aid of linear programming. The increasing practical use of linear programming since 1947 is linked with the rise of electronic computers, and 1947 itself is important as the year in which George B. Dantzig developed the first, and so far the best, algorithm for the solution of the general linear programming problem: the simplex method.

Unfortunately the simplex method, as it is used on computers today, does not handle every problem with the success which it deserves. At times, with no forewarning, computer programs which implement the simplex method will produce unreasonable results from reasonable data. In such cases curses are muttered against a mysterious force called "round-off error", and a battery of *ad hoc* panaceas are employed to attempt a cure. Sometimes, in hopes of forestalling problems from the outset, a variety of heuristic tests and purifications may be built into a computer program to sniff at suspected indicators of round-off error and attempt recoveries during the computation. But all of the medicines employed have the aspect of folk remedies. They are largely in the domain of oral tradition; they are rarely published, and their efficacy has never been proved.

The literature concerning the effects of round-off errors upon the simplex method is very small (we have come across only papers [3], [8], [10], and [15] listed in the bibliography), and it contains no rigorous results. This state of affairs is unfortunate and unnecessary.

James H. Wilkinson first published the tools for investigating round-off error effects in linear algebraic computations in 1959, [11], and they have been well-known to numerical analysts since then. Numerical analysts, however, have not made much use of linear programming. The subject is just beginning to be recognized in the numerical analytical world as a piece of practical machinery (e.g., Rabinowitz [9]). The chief users of linear programming up to now, on the other hand, could not be expected to be familiar with the literature on round-off errors in linear algebraic computations (e.g., Chartres and Geuder [2], and Wilkinson [11], [12], [13], [14]). Thus a proper investigation of round-off errors in linear programming has never been made. This thesis, we hope, will serve as a first step toward changing this.

It is important to separate the abstract expression of a computational algorithm from the realization of that algorithm on a computer. In Chapter 2 we present the simplex method in an abstract setting, and in Chapter 3 we describe the main details of the method's more common implementations on computers. We show in Chapter 4 that round-off errors can, indeed, have disasterous effects upon the common simplex-method computer programs. This is not due to any flaw in the simplex method. It is rather due to the use of Gauss-Jordan elimination without pivot selection which characterizes the standard computer implementations of the simplex method. In Chapter 4 we indicate that no simple _a priori_ inspection of the data will yield a bound on the effects of round-off errors in programs which employ Gauss-Jordan elimination but do not permit the pivots to be selected arbitrarily.

Chapters 6 and 8 present alternative computer implementations of the simplex method for which pivot selection is flexible, and these implementations are shown in Chapters 7 and 8 to be numerically stable. That is, the effects of round-off errors in these methods can be bounded in a simple *a priori* fashion from the data.

In each of the simplex-method implementations presented in this thesis the possibility of monitoring round-off errors during the computation exists. The discussion of this possibility is contained in Chapters 5, 9, and 10, wherein *a posteriori* error bounds and their employment in simplex-method computation is covered.

In rough terms, carrying out the simplex method entails the repetitive solution of three linear-equation systems

$$\begin{cases} Bu = b \\ B^T\pi = \gamma \\ By = A_s \end{cases} .$$

(The notation we use in dealing with the simplex method holds closely to that used by Dantzig in [4].) At each repetition the vectors u, π, and y are inspected, the *basic matrix* B is modified as a result of the inspection and a new set of vectors is produced. It is the main effect of round-off errors that only approximate vectors $u + \delta u$, $\pi + \delta\pi$, and $y + \delta y$ are produced from the computations. Assuming that bounds can be placed on the norms of the errors $\|\delta u\|$, $\|\delta\pi\|$, and $\|\delta y\|$, then Chapter 10 discusses the use of these bounds during simplex-method computation to make allowances for round-off effects. (Throughout the thesis the double bars $\| \;\|$ denote the L_2 matrix and vector norms. That is,

$$\|v\| = \sqrt{\sum_i v_i^2}$$

for any vector v, and

$$\|M\| = \max_{x \neq 0} \frac{\|Mx\|}{\|x\|} = \sqrt{\lambda_{\max}(MM^T)}$$

for any square matrix M .)

For the standard, explicit-inverse implementations of the simplex method, Chapter 5 covers the computation of error-norm bounds. The bounds themselves are given by the equation

$$\|\delta v\| \leq \frac{1}{1-\|E\|} (\|E\|\|v+\delta v\|+\|\Phi\|\|g\|)$$

appearing in mid-chapter, where δv represents any of the error vectors, and g represents any of the right-hand vectors. A bound for $\|\Phi\|$ is presented, and the last half of the chapter is addressed to the problem of efficiently computing E, or at least an upper bound for $\|E\|$.

The other two implementations presented in the thesis use forms of triangular decompositions for matrices as substitutes for the inverse of the basic matrix. The standard LU implementation, described in Chapter 3, dec mposes the basic matrix into triangular factors L and U . The modified LU implementation, described in Chapter 6, decomposes the basic matrix into a triangular factor L, a product of simple matrices G, and a triangular factor U . It is shown in Chapters 7 and 8 that, for both implementations, the computed solution $v + \delta v$ to the linear system

$$Bv = g$$

satisfies

$$(B+\mathcal{E})(v+\delta v) = g \ .$$

Both of these chapters also address themselves to the problem of bounding $\|\mathcal{E}\|$. For the modified LU implementation the results are summarized in the last four pages of Chapter 7. For the standard LU implementation the bound is given in Chapter 8 as

$$\|\mathcal{E}\| \leq m[3\tilde{\varepsilon}_1+2m(m+3)\tilde{\varepsilon}_2] \max_{i,j} |u_{ij}| \ ,$$

where the quantities $\tilde{\varepsilon}_1$ and $\tilde{\varepsilon}_2$ are defined during the development of this bound, and m is of the order B .

The error vector δv is shown in Chapter 9 to satisfy

$$\delta v = - B^{-1}\mathcal{E}(v+\delta v) \ ,$$

so that

$$\|\delta v\| \leq \|B^{-1}\|\|\mathcal{E}\|\|v+\delta v\| \ .$$

Since $\|v+\delta v\|$ is known and a bound for $\|\varepsilon\|$ has been provided by previous chapters, Chapter 9 addresses itself mainly to the problem of finding an upper bound for $\|B^{-1}\|$ easily from the information which is at hand. This information is, roughly, the trace and determinant of BB^T.

The trace, d, of BB^T is readily computed, but only an approximation to $\det(BB^T)$ can be made. The analysis of Chapters 7 and 8 show that the computed decompositions of B used in the standard and the modified LU implementations actually satisfy

$$B + \delta B = \text{decomposition}$$

because of round-off er.or effects, where a bound for $\|\delta B\|$ is given. Thus, the quantity

$$p = \det([B+\delta B][B+\delta B]^T)$$

is at hand. It is shown in Chapter 9 that the best upper bound on $\|B^{-1}\|$ which can be made using p and d is

$$\sqrt{\frac{1}{q}}$$

where q satisfies

$$q = p\left(\frac{m-1}{d-mK-q}\right)^{m-1} ,$$

and where

$$K \geq \|\delta BB^T + B\delta B^T + \delta B\delta B^T\| \quad .$$

Finally, in Chapter 11 a comparison of the modified and standard LU implementations of the simplex method is made with a representative explicit-inverse implementation in terms of speed of operation. This is done by counting arithmetic operations needed by each to produce u, π, and y and to accomodate the changes in B which follow. The results are reviewed on the last page of the chapter. In particular we find that the modified LU implementation, which we have shown to be not subject to round-off error disasters, is competitive in speed with standard simplex-method implementations.

2. Review of the Simplex Method of Linear Programming

To begin with, we will briefly cover the basic concepts and terminology of linear programming and of the simplex method. Everything to be mentioned is covered in greater detail in Dantzig [4] and in Hadley [6].

A linear programming problem is, by definition, one which can be put in the following form:

$$\begin{cases} \text{maximize the } \underline{\text{objective}}\ \underline{\text{function}}: \quad z = c^T x \\ \text{subject to the } \underline{\text{constraints}} \quad Ax = b \quad \text{and} \quad x \geq 0 \,. \end{cases}$$

A is an $m \times n$, real matrix having full row rank, and $m < n$. The vectors b and c are real and have dimensions m and n respectively. For convenience we also require that $b \geq 0$, but this is not essential to the problem.

In the standard terminology any vector x which satisfies the constraints is a <u>feasible solution</u>, and, if it maximizes the objective function, it is an <u>optimal feasible solution</u>.

Let $A_{\nu_1}, \ldots, A_{\nu_m}$ be m linearly independent columns from the matrix A. If x is a feasible solution having $x_j = 0$ for all $j \notin \{\nu_1, \ldots, \nu_m\}$, then x is called a <u>basic feasible solution</u>, and the columns $A_{\nu_1}, \ldots, A_{\nu_m}$ are called the <u>basic vectors</u> (or simply the <u>basis</u>) associated with x. Any $m \times m$ matrix composed of these columns is a <u>basic matrix</u> corresponding to x. If $x_i = 0$ for some $i \in \{\nu_1, \ldots, \nu_m\}$, then x is a <u>degenerate</u> basic feasible solution.

The set of all feasible solutions to a linear programming problem is convex, and the basic feasible solutions comprise precisely the set of extreme points. The existence of any feasible solutions implies the existence of at least one basic feasible solution. And, if the objective function is bounded from above on the set of feasible solutions, at least one of the basic feasible solutions is optimal. (In fact, several of the basic feasible solutions can be optimal. A given linear programming problem need not have a unique solution.)

It is upon bases and basic feasible solutions that Dantzig's simplex method is grounded. The simplex method is an algorithm which searches from basic feasible solution to basic feasible solution, seeking an optimal solution. The search is designed so that each basic feasible solution encountered has a greater value of the associated objective function than its predecessor. The search proceeds in cycles. Let x be a nondegenerate basic feasible solution. The <u>simplex-method cycle</u> produces one of the following from x:

I. a basic feasible solution x' from which

$$c^T x < c^T x' ;$$

II. an indication that x is optimal;

III. an indication that the objective function is unbounded from above on the set of feasible solutions.

Degenerate basic feasible solutions do not yield quite such reasonable outcomes from the simplex-method cycle. However, there are standard ways of adjusting a given linear programming problem so that degeneracies which are discovered are wiped away, which fact will permit us the liberty of ignoring degeneracies from this point forward.

To each basic feasible solution there is a corresponding basis of m linearly independent columns taken from the n columns of A. And outcome (I) of the simplex-method cycle guarantees that the simplex method will never encounter any basic feasible solution more than once. Therefore, starting from any particular basic feasible solution, the simplex method cannot be carried through more than

$$\frac{n!}{m!(n-m)!}$$

cycles on a given linear programming problem before terminating. In practice one commonly finds that the number of cycles taken before termination is only a small multiple of m (Dantzig [4, p. 160]).

The mechanism for producing x' from x by the simplex-method cycle is built upon basic matrices. If $A_{\nu_1}, \ldots, A_{\nu_m}$ is the basis corresponding to x, then consider the basic matrix

$$B = [A_{\nu_1}, \ldots, A_{\nu_m}] .$$

Let

$$u = [x_{\nu_1}, \ldots, x_{\nu_m}]^T$$

be the vector of the nonzero components of x. Then

$$Bu = b .$$

To carry out a simplex-method cycle, one begins with B and proceeds as follows:

1. Solve $Bu = b$.
2. Let $\gamma = [c_{\nu_1},\ldots, c_{\nu_m}]^T$, the vector composed from c by selecting those components corresponding to the basis.
3. Solve $B^T\pi = \gamma$.
4. For each column A_j of A not in the basis compute

$$\bar{c}_j = \pi^T A_j - c_j ,$$

   and choose any index s for which $\bar{c}_s < 0$.
5. Solve $By = A_s$.
6. Let r be an index such that

$$\frac{u_r}{y_r} = \min_{y_i>0} \frac{u_i}{y_i} .$$

7. Drop the vector A_{ν_r} from the basis and add the vector A_s . The resulting set of m vectors is a new basis, whose associated basic feasible solution is the vector x' . The associated basis matrix B' can be constructed with whatever ordering of columns is most convenient.

The cycle cannot be carried past step 3 if all $\bar{c}_j$ are nonnegative, in which case x is optimal. And the cycle cannot be carried out past step 5 if no component of the y vector is positive, in which case the objective function is unbounded on the set of feasible solutions.

The change in the value of the objective function

$$z' - z = c^T(x' - x)$$

is given by

$$\frac{u_r}{y_r}(c_s - \pi^T A_s) = -\frac{u_r}{y_r}\bar{c}_s .$$

Thus it makes some sense to choose s such that

$$\bar{c}_s = \min_j \bar{c}_j ,$$

making $\bar{c}_s$ as largely negative as possible, in hopes of making as great a change in the objective function as possible.

To solve a given linear programming problem by the simplex method, a basic feasible solution must first be found. If none exists, the problem is <u>infeasible</u>. The task of finding the initial basic feasible solution can be formulated as an auxiliary linear programming problem, one for which a starting basis is known. Hence, the simplex method is generally carried out as a two-phase process, each phase of which consists of repeated applications of the simplex-method cycle. The main part of this thesis will treat with the computational details of carrying out the simplex-method cycle on an automatic digital computer and with the round-off errors incurred during the computation.

3. Standard Computer Implementations of the Simplex Method

The major price to be paid in carrying out the simplex-method cycle is incurred by solving the linear systems

$$\begin{cases} Bu = b \\ B^T\pi = \gamma \\ By = A_s \end{cases} .$$

If these systems were presented in isolation, we would expect to spend an order of m^3 arithmetic operations on a computing machine to get their solutions. In the context of the simplex method, however, this much work is not usually required. The basic matrix B used in one execution of the simplex-method cycle is constructed from all but one of the columns which formed the basic matrix used in the preceding execution of the simplex-method cycle. The vector b remains constant throughout, and the vector γ changes in only one component at each cycle. These observations provide the opportunity for significant savings in computational effort.

For example, in representative computer implementations of the simplex method the matrix of order $m + 1$

$$D = \begin{array}{|c|c|} \hline 1 & -\gamma^T \\ \hline 0 & B \\ \hline \end{array}$$

is constructed. Since

$$D^{-1} = \begin{array}{|c|c|} \hline 1 & \pi^T \\ \hline 0 & B^{-1} \\ \hline \end{array} \; ,$$

it can be verified that

$$\left[\frac{z}{u}\right] = D^{-1}\left[\frac{0}{b}\right],$$

and that

$$\left[\frac{\bar{c}_s}{y}\right] = \left[\frac{\pi^T A_s - c_s}{y}\right] = D^{-1}\left[\frac{-c_s}{A_s}\right].$$

Let us number by zero the initial components of the vectors, and the initial rows and columns of the matrices in what follows directly below. This will unify the indices used in this chapter with those used throughout the rest of the thesis.

Suppose an execution of the simplex-method cycle ends in outcome (I). If we let

$$D' = \begin{array}{|c|c|} \hline 1 & -(\gamma')^T \\ \hline 0 & B' \\ \hline \end{array}$$

by replacing the r th column of D with

$$\left[\frac{-c_s}{A_s}\right],$$

then it can be verified that $(D')^{-1}$ is obtainable from D^{-1} by the application of the Gauss-Jordan elimination which reduces the vector

$$t = \left[\frac{\bar{c}_s}{y}\right]$$

to the r th unit vector. That is, if

$$T = \begin{array}{|ccc|c|ccc|c}
\multicolumn{3}{c}{} & \multicolumn{1}{c}{r} & \multicolumn{3}{c}{} & \\
\hline
1 & & & -t_0/t_r & & & & \\
 & \ddots & & \vdots & & & & \\
 & & 1 & -t_{r-1}/t_r & & & & \\
\hline
 & & & 1/t_r & & & & r \\
\hline
 & & & -t_{r+1}/t_r & 1 & & & \\
 & & & \vdots & & \ddots & & \\
 & & & -t_m/t_r & & & 1 & \\
\hline
\end{array}$$

$$= [I - \frac{1}{t_r}(t-e^{(r)})e^{(r)^T}],$$

where $e^{(r)}$ is the r th unit vector, then

$$(D')^{-1} = TD^{-1} .$$

Similarly we can update the vector u, rather than solve a system of equations for u :

$$\left[\frac{z}{u}\right] = T\left[\frac{z}{u}\right].$$

In this manner the simplex-method cycle can be carried out on a computer with only an order of m^2 arithmetic operations.

If D^{-1} is explicitly computed in an implementation of the simplex method, is kept in computer storage as an $m \times m$ matrix, and is updated by Gauss-Jordan elimination at each cycle, the implementation is said to use the <u>explicit inverse</u>. Alternatively, D^{-1} can be expressed as a product of transformations which reduces D to the identity matrix. At the end of each execution of the simplex-method cycle, a Gauss-Jordan transformation can simply be added to the product so that

$$D^{-1} = T^{(\mu)}T^{(\mu-1)} \dots T^{(1)} .$$

Implementations of the simplex method built along these lines are said to use the <u>product form of the inverse</u>. They are used when the data matrix A is sparse and has its nonzero entries in a regular pattern which permits the use of transformations $T^{(i)}$ having a particularly simple form.

The basic computational process in implementations of either type is the Gauss-Jordan elimination applied to a vector, either a column of D^{-1}, the vector

$$\begin{bmatrix} z \\ \hline u \end{bmatrix}$$

or, in case the product form of the inverse is used, to the vector

$$\begin{bmatrix} c_s \\ \hline A_s \end{bmatrix} .$$

Hence, we will want to study this process as it is carried out in floating-point arithmetic on contemporary computing machines.

4. Round-off Error Analysis of Gauss-Jordan Elimination

We will denote the computed value of an arithmetic expression $\mathcal{E}$ obtained by floating-point operations on a computer by

$$fl(\mathcal{E}) .$$

This value is a function not only of $\mathcal{E}$ but also of the computer employed and the algorithm used for the evaluation. Following Wilkinson [13], we know that on reasonable computers, so long as overflow or underflow do not occur,

$$\begin{cases} fl(a+b) = a(1+\eta_1) + b(1+\eta_2) \\ fl(a-b) = a(1+\eta_3) - b(1+\eta_4) \\ fl(a\times b) = a \times b \times (1+\eta_5) \\ fl(a/b) = (a/b) \times (1+\eta_6) \end{cases}$$

for any two machine numbers a and b, where the quantities $|\eta_i|$ are bounded by some fixed, small, positive number. In fact we will assume that our computations are carried out on a machine offering a <u>normal-precision</u> floating-point arithmetic, for which

$$|\eta_i| \leq \varepsilon_1 ,$$

and an <u>extended-precision</u> floating-point arithmetic, for which

$$|\eta_i| \leq \varepsilon_2 ,$$

where ε_2 is a few orders of magnitude smaller than ε_1. This is usually accomplished by carrying out floating-point arithmetic in base τ arithmetic (τ = 2, 8, 10, 16 are common), carrying σ_1 significant figures in normal arithmetic and σ_2 significant figures in extended arithmetic (usually with $\sigma_2 \geq 2\sigma_1$). If that is the case, one may take

$$\varepsilon_1 = \tau^{1-\sigma_1}$$

and

$$\varepsilon_2 = \tau^{1-\sigma_2} .$$

A <u>normal-precision machine number</u> is a number which may be represented exactly on the computer to the precision of normal arithmetic. A similar definition holds for <u>extended-precision machine numbers</u>.

Suppose we transform the vector $(w_1, \ldots, w_\ell)$ into the vector $(v_1, \ldots, v_\ell)$ by applying Gauss-Jordan elimination, obtaining the multipliers from the vector $(p_1, \ldots, p_\ell)$, with p_β used as the pivot $(1 \leq \beta \leq \ell)$. The p's, v's, and w's are all to be normal-precision machine numbers.

Thus,

$$v_\beta = f\ell(w_\beta/p_\beta) ,$$
$$v_i = f\ell(w_i - p_i w_\beta/p_\beta) , \quad \text{for } i \neq \beta .$$

Or

$$v_\beta = (w_\beta/p_\beta)(1+\eta) ,$$
$$v_i = w_i(1+\varphi_i) - (p_i w_\beta/p_\beta)(1+\eta)(1+\mu_i)(1+\rho_i), \quad \text{for} \quad i \neq \beta .$$

This means that, letting $\delta w_\beta = \eta w_\beta$,

$$v_\beta = \frac{1}{p_\beta}(w_\beta + \delta w_\beta) .$$

And, for $i \neq \beta$,

$$v_i = \{w_i - \frac{p_i}{p_\beta} w_\beta\}$$
$$+ \{[w_i\varphi_i - \frac{p_i}{p_\beta} w_\beta(\mu_i+\rho_i+\mu_i\rho_i+\eta\mu_i+\eta\rho_i+\eta\mu_i\rho_i)] - \frac{p_i}{p_\beta}\eta w_\beta\} .$$

Or, letting

$$\delta w_i = w_i\varphi_i - \frac{p_i}{p_\beta} w_\beta(\mu_i+\rho_i+\mu_i\rho_i+\eta\mu_i+\eta\rho_i+\eta\mu_i\rho_i) ,$$

we have

$$v_i = (w_i + \delta w_i) - \frac{p_i}{p_\beta}(w_\beta + \delta w_\beta) .$$

Thus, if

$$P = [I - \frac{1}{p_\beta}(p - e^{(\beta)})e^{(\beta)^T}] ,$$

it has been established that

$$v = fl(Pw) = P(w+\delta w) .$$

That is, the round-off errors have the effect of making the computed vector v be an exact transform via the given Gauss-Jordan elimination of a perturbation on the vector w .

How large can the perturbation δw be? The answer is disturbing. Unless restrictions are placed upon the pivot p_β, the components of δw cannot be bounded. We have, for $i \neq \beta$:

$$|\delta w_i| \leq |w_i||\varphi_i| + \frac{|p_i|}{|p_\beta|} |w_\beta||\mu_i+\rho_i+\mu_i\rho_i+\eta\mu_i+\eta\rho_i+\eta\mu_i\rho_i| ,$$

or

$$|\delta w_i| \leq |w_i|\varepsilon_1 + \frac{|p_i|}{|p_\beta|} |w_\beta|(2\varepsilon_1 + 3\varepsilon_1^2 + \varepsilon_1^3) .$$

And so, if $|p_\beta|$ is made small enough relative to the magnitude of some one of the other components of the vector p, the right-hand side of the latter inequality can be made arbitrarily large for the corresponding value of i .

To see a trivial example of what this means, suppose that we have a computing machine at our disposal which performs decimal arithmetic with three figures of precision in normal floating-point calculations. The result of each basic arithmetic operation, however, is to be computed to six figures of accuracy before being rounded to the nearest three significant figures. This means that

$$\epsilon_1 = \tfrac{1}{2}10^{-2} .$$

Let

$$w = (1,1) , \quad p = (10^{-4},1) , \quad \text{and} \quad \beta = 1 .$$

Then

$$v_1 = fl(1/10^{-4}) = 10^4, \quad \text{and}$$
$$v_2 = fl(1-1\times1/10^{-4}) = -10^{-4} .$$

That is,

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 10^4 \\ -10^4 \end{bmatrix} = \begin{bmatrix} 10^4 & 0 \\ -10^4 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} .$$

So $\delta w_2 = -w_2$. A perturbation has been made in one of the components of w which is as large as the component itself. If there was any useful information in the vector w initially, it has been lost.

It is easy to concoct examples wherein relatively small pivots arise during execution of the simplex method. E.g., take the data

$$c^T = [1 \quad 1 \quad 2]$$
$$A = \begin{bmatrix} 1 & 0 & \gamma \\ 0 & 1 & 1 \end{bmatrix} , \quad b = \begin{bmatrix} \gamma/2 \\ 1 \end{bmatrix} ,$$

where Ψ is a small positive number less than one. The first two columns of A will serve as an initial basis, giving

$$D = \begin{bmatrix} 1 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$D^{-1} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

and

$$\begin{bmatrix} z \\ u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 1 + \Psi/2 \\ \Psi/2 \\ 1 \end{bmatrix}.$$

Since $\bar{c}_3 = \pi^T A_3 - c_3 = \Psi - 1$, which is less than zero, the cycle proceeds. In the notation of Chapter 2, $s = 3$, and we must compute

$$\begin{bmatrix} \bar{c}_3 \\ y_1 \\ y_2 \end{bmatrix} = D^{-1} \begin{bmatrix} -2 \\ \Psi \\ 1 \end{bmatrix} = \begin{bmatrix} \Psi - 1 \\ \Psi \\ 1 \end{bmatrix}$$

Both components of y are positive.

$$\frac{u_1}{y_1} = \frac{1}{2} \quad \text{and} \quad \frac{u_2}{y_2} = 1 \ .$$

So $\dfrac{u_1}{y_1} = \min$, or $r = 1$ in the notation of Chapter 2. Hence, the basic vector

$$A_3 = \begin{bmatrix} \Psi \\ 1 \end{bmatrix}$$

must be exchanged for the basic vector

$$A_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} .$$

Thus all necessary updating is to be carried out by the Gauss-Jordan elimination composed from the vector

$$\begin{bmatrix} \bar{c}_3 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \Psi - 1 \\ \Psi \\ 1 \end{bmatrix}$$

using $y_1 = \Psi$ as the pivot, which is small relative to the other two components.

The implication of all of this for standard implementations of the simplex method is that, since the relative sizes of the pivots which arise during computation cannot be easily predicted, no bound, simply expressed in terms of the data, can be made <u>a priori</u> on the perturbations which will be caused by round-off errors.

5. Error Bounds for Linear-system Solutions Found by Explicit Inverse

Though it is not possible *a priori* to bound round-off perturbations during application of the simplex method using computer implementations such as those just discussed, we can compute round-off error bounds in an *a posteriori* fashion. We will do that here in connection with explicit-inverse implementations. Product-form implementations can be treated in a similar fashion.

Consider the following problem: if P is a given, nonsingular matrix of order ℓ, if g is an ℓ vector, and if the matrix Q is accepted as an approximate inverse to P, how much error is made in taking

$$fl(Qg)$$

as the solution to

$$Pv = g\ ?$$

Let $v = P^{-1}g$ and $v + \delta v = fl(Qg)$. Each component of $v + \delta v$ is formed as an inner product:

$$v_i + \delta v_i = fl(\sum_{j=1}^{\ell} q_{ij}g_j) .$$

The calculation of an inner product is so basic to computational linear algebra that, in order to be as accurate as possible, it is a frequent practice to form the products and partial sums in extended-precision arithmetic. Only the full sum is reduced to normal precision (Wilkinson [13, p. 23]). That is,

$$\begin{cases} t^{(0)} = 0 \\ t^{(j)} = fl(t^{(j-1)}+q_{ij}g_j) = t^{(j-1)}(1+\alpha_j) + q_{ij}g_j(1+\beta_j)(1+\eta_j) \end{cases}$$

$$\text{for } j = 1, \ldots, \ell ,$$

where $|\alpha_j|$, $|\beta_j|$, and $|\eta_j|$ are all bounded by ϵ_2. Only $t^{(\ell)}$ is reduced to normal precision, giving:

$$v_i + \delta v_i = t^{(\ell)}(1+\rho_i) ,$$

where $|\rho_i| \le \epsilon_1$. Therefore

$$v_i + \delta v_i = \sum_{j=1}^{\ell} [q_{ij}(1+\rho_i)(1+\beta_j)(1+\eta_j) \prod_{k=j+1}^{\ell} (1+\alpha_k)]g_j ,$$

or

$$v_i = \delta v_i = \sum_{j=1}^{\ell} (q_{ij}+\varphi_{ij})g_j ,$$

where

$$\varphi_{ij} = q_{ij}[(1+\rho_i)(1+\beta_j)(1+\eta_j) \prod_{k=j+1}^{\ell} (1+\alpha_k)-1]$$

Thus, in matrix form,

$$v + \delta v = (Q+\Phi)g ,$$

where

$$|\varphi_{ij}| \leq |q_{ij}|\{\epsilon_1 + (1+\epsilon_1)\frac{\ell-j+2}{\ell}[\exp(\frac{\ell\epsilon_2}{1-\epsilon_2})-1]\} .$$

The bound

$$\epsilon_1 + (1+\epsilon_1)\frac{\ell-j+2}{\ell}[\exp(\frac{\ell\epsilon_2}{1-\epsilon_2})-1]$$

$$\geq |(1+\rho_i)(1+\sigma_j)(1+\eta_j)\prod_{k=j+1}^{\ell}(1+\alpha_k)-1|$$

will be established in Chapter 7.

Let the residual matrix

$$E = I - QP$$

be defined. We will demand that Q be such a good approximation to the inverse of P that $\|E\| < 1$. Then the inverse of $I - E$ will exist, and

$$\|(I-E)^{-1}\| \leq \frac{1}{1-\|E\|} .$$

Therefore

$$\begin{aligned} v + \delta v &= Qg + \Phi g \\ &= QPv + \Phi g \\ &= QP(v+\delta v) - QP\delta v + \Phi g , \end{aligned}$$

which gives

$$\delta v = (I-E)^{-1}[\Phi g - E(v+\delta v)] ,$$

or

$$\|\delta v\| \le \frac{1}{1-\|E\|} (\|E\|\|v+\delta v\| + \|\Phi\|\|g\|) .$$

The cost of computing the residual matrix E is, in ordinary situations, far too great to pay for simply putting a bound on $\|\delta v\|$. An order of ℓ^3 arithmetic operations is needed to form E. That is essentially the same amount of work as is required to invert the matrix P. There are far more lucrative ways to expend one's energy. The often-used technique of iterative refinement (Wilkinson [13, p. 130], Forsythe and Moler [5, Chapter 13]), for example, can correct $v + \delta v$ to a point at which $\|\delta v\| \approx \varepsilon_1$ and can be carried out in an order of ℓ^2 operations. But, in the context of the simplex method, the computation of this error bound can become quite economical.

Suppose P' is obtained from P by replacement of the r^{th} column $Pe^{(r)}$ with the vector w :

$$P' = P - (Pe^{(r)} - w)e^{(r)^T} .$$

Let the vector t be given by

$$t = fl(Qw) ,$$

and compute Q', an approximation to $(P')^{-1}$, by applying to Q the Gauss-Jordan elimination which reduces t to the rth unit vector:

$$Q' = fl\{[I - \frac{1}{t_r}(t-e^{(r)})e^{(r)^T}]Q\} .$$

(This imitates the situation in typical explicit-inverse implementations of the simplex method.) Apply the results of Chapter 4 columnwise to Q to get:

$$Q' = [I - \frac{1}{t_r}(t-e^{(r)})e^{(r)^T}][Q+\Psi] ,$$

where for all i and j :

$$|\psi_{ij}| \leq \begin{cases} |\frac{1}{t_r}| \, |q_{rj}| \, \varepsilon_1 & \text{if } i = r \\ |q_{ij}| \, \varepsilon_1 + |\frac{t_i}{t_r}| \, |q_{rj}| \, (2\varepsilon_1+3\varepsilon_1^2+\varepsilon_1^3) & \text{if } i \neq r . \end{cases}$$

For convenience let

$$T = [I - \frac{1}{t_r}(t-e^{(r)})e^{(r)^T}] .$$

Then

$$\begin{aligned} E' &= I - Q'P' \\ &= I - T(Q+\Psi)P' \\ &= I - TQP' - T\Psi P' . \end{aligned}$$

But

$$I - TQP' = I - [I - \frac{1}{t_r} (t-e^{(r)})e^{(r)^T}]Q[P-(Pe^{(r)}- w)e^{(r)^T}]$$

$$= I - [QP - \frac{1}{t_r} (t-e^{(r)})e^{(r)^T}QP-Q(Pe^{(r)}- w)e^{(r)^T}$$

$$+ \frac{1}{t_r} (t-e^{(r)})e^{(r)^T}Q(Pe^{(r)}- w)e^{(r)^T}]$$

$$= E - [\frac{1}{t_r} (t-e^{(r)})e^{(r)^T}Q(Pe^{(r)}- w)e^{(r)^T}$$

$$- \frac{1}{t_r} (t-e^{(r)})e^{(r)^T}QP-Q(Pe^{(r)}- w)e^{(r)^T}] .$$

We make thc substitution

$$t = f\ell(Qw) = (Q+\Omega)w ,$$

for which

$$|\omega_{ij}| \leq \{\epsilon_1 + (1+\epsilon_1) \frac{\ell-j+2}{\ell} [\exp(\frac{\ell\epsilon_2}{1-\epsilon_2})-1]\} |q_{ij}| ,$$

and the substitution

$$QP = I - E$$

to obtain

$$\begin{aligned} I - TQP' = E - \{ & \frac{1}{t_r} (t-e^{(r)})(e^{(r)^T} e^{(r)}) e^{(r)^T} \\ & - \frac{1}{t_r} (t-e^{(r)}) e^{(r)^T} E e^{(r)} e^{(r)^T} \\ & - \frac{1}{t_r} (t-e^{(r)})(e^{(r)^T} t) e^{(r)^T} \\ & - \frac{1}{t_r} (t-e^{(r)}) e^{(r)^T} \\ & + \frac{1}{t_r} (t-e^{(r)}) e^{(r)^T} E - e^{(r)} e^{(r)^T} \\ & + E e^{(r)} e^{(r)^T} + t e^{(r)^T} \} \\ & + \Omega w e^{(r)^T} - \frac{1}{t_r} (t-e^{(r)}) e^{(r)^T} \Omega w e^{(r)^T} . \end{aligned}$$

A couple of simplifications are possible:

$$\frac{1}{t_r} (t-e^{(r)})(e^{(r)^T} e^{(r)}) e^{(r)^T} = \frac{1}{t_r} (t-e^{(r)}) e^{(r)^T} ,$$

since $e^{(r)^T} e^{(r)} = 1$,

and

$$\frac{1}{t_r}\,(t-e^{(r)})(e^{(r)^T}t)e^{(t)^T} = (t-e^{(r)})e^{(r)^T} ,$$

$$\text{since}\quad e^{(r)^T}t = t_r .$$

Therefore

$$I = TQP' = E - \{ \frac{1}{t_r}\,(t-e^{(r)})e^{(r)^T}E(I-e^{(r)}e^{(r)^T})$$

$$+ Ee^{(r)}e^{(r)^T}\} + T\Omega we^{(r)^T}$$

$$= [I - \frac{1}{t_r}\,(t-e^{(r)})e^{(r)^T}]E(I-e^{(r)}e^{(r)^T})$$

$$+ T\Omega we^{(r)^T} .$$

So

$$E' = T[E(I-e^{(r)}e^{(r)^T}) - \Omega we^{(r)^T} - \Psi P'] .$$

And therefore,

$$\|E'\| \leq \|T\|(\|E\|+\|\Omega\|\|w\|+\|\Psi\|\|P'\|) .$$

If upper bounds for $\|E\|$, $\|\Omega\|$, and $\|\Psi\|$ are at hand, an upper bound for $\|E'\|$ is thereby available. And only an upper bound for the residual matrix norm is needed to bound the error $\|\delta v\|$.

Returning to the notation and concepts of Chapters 2 and 3, it is clear that the above results can be employed during the execution of the simplex method cycle to bound the computational error incurred in finding u, y, π, z, and the $\bar{c}_i$. In a later chapter, in connection with another implementation of the simplex method, we will consider the employment of such error bounds to provide automatic tolerance control. The techniques presented there can easily be imitated with the explicit-inverse implementation of the simplex method.

6. The Modified LU Implementation of the Simplex Method

Returning to the notation of Chapter 2, we let B be the $m \times m$ nonsingular basic matrix and concern ourselves with solving the linear systems

$$Bu = b\ , \quad B^T\pi = \gamma\ ; \quad By = A_s\ .$$

Suppose we effect this by decomposing B into the product of a lower and an upper triangular matrix. Such a decomposition is always possible provided that, at the least, row permutations of B are permitted. Column permutations could be added, but, in the context of the simplex method in which we work, it will not be convenient to permute the columns of each basic matrix we encounter arbitrarily.

Thus we have

$$\Pi B = LU$$

for some permutation matrix Π, upper triangular matrix U and lower triangular matrix L, where we further demand that all diagonal elements of L be $+1$.

A given linear system

$$Bv = q$$

which we must solve is handled by solving the triangular systems

$$Lt = \Pi q ,$$

$$Uv = t .$$

The round-off error analysis of the decomposition and solution of the triangular systems is given in Wilkinson [13]. The best error bounds are obtained if Π is chosen so that all elements of L are bounded in magnitude by unity. The strategy of constructing Π during the computation of the decomposition has come to be called "partial pivoting" or "row pivoting".

At the beginning of the simplex method cycle B has the form

$$B = [A_{\nu_1}, \ldots, A_{\nu_m}] .$$

At the end of the cycle we may be required to construct a new basic matrix $B^{(1)}$ from the columns

$$A_{\nu_1}, \ldots, A_{\nu_{r_1-1}}, A_{\nu_{r_1+1}}, \ldots, A_{\nu_m}, A_{s_1} .$$

And this is to be done in such a way that the transition from the decomposition of B to one for $B^{(1)}$ is particularly easy and computationally stable.

Consider, then, the basic matrix

$$B^{(1)} = [A_{\nu_1}, \ldots, A_{\nu_{r_1-1}}, A_{\nu_{r_1+1}}, \ldots, A_{\nu_m}, A_{s_1}] ;$$

i.e., let $B^{(1)}$ be obtained from B by dropping the r_1th column, shifting all subsequent columns one position to the left, and appending A_{s_1} on the right. Therefore, columnwise:

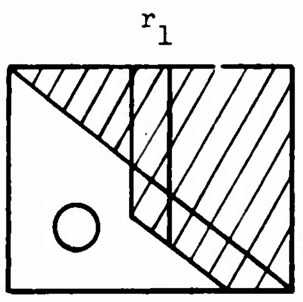
$$L^{-1}\Pi B^{(1)} = [L^{-1}\Pi A_{\nu_1}, \ldots, L^{-1}\Pi A_{\nu_{r_1-1}},$$

$$L^{-1}\Pi A_{\nu_{r_1+1}}, \ldots, L^{-1}\Pi A_{\nu_m}, L^{-1}\Pi A_{s_1}]$$

$$= [U_1, \ldots, U_{r_1-1}, U_{r_1+1}, \ldots, U_m, L^{-1}\Pi A_{s_1}]$$

$$= H^{(1)} ,$$

where the U_i are the columns of U. $H^{(1)}$ is a matrix of the form



;

that is, an upper Hessenberg matrix with zeros below the diagonal in the first r_1-1 columns. We point out that the vector $L^{-1}\Pi A_{s_1}$ was produced as a byproduct during the computation of the vector y. Hence $H^{(1)}$ can be constructed without any computational effort.

$H^{(1)}$ can be reduced to an upper triangular matrix $U^{(1)}$ by using Gaussian eliminations to zero out the subdiagonal elements in columns r_1 through m. The partial pivoting technique is carried out by making a decision at each elimination step whether or not to interchange two adjacent rows. Thus, $U^{(1)}$ is gotten from $H^{(1)}$ by applying a sequence of simple transformatins on the left:

$$U^{(1)} = \Gamma_{m-1}^{(1)}\Pi_{m-1}^{(1)} \cdots \Gamma_{r_1}^{(1)}\Pi_{r_1}^{(1)}H^{(1)} ,$$

where each $\Gamma_i^{(1)}$ has the form

$$\begin{array}{c|c|c|c|c|}
 & & i & i+1 & \\ \hline
 & 1 \ddots 1 & & & \\ \hline
i & & 1 & & \\ \hline
i+1 & & -g_i^{(1)} & 1 & \\ \hline
 & & & & 1 \ddots 1 \\ \hline
\end{array} ,$$

and each $\Pi_i^{(1)}$ is either the identity matrix or the identity with the i^{th} and $i+1^{st}$ rows exchanged, the choice being made so that $|g_i^{(1)}| \leq 1$.

Thus

$$\Pi B^{(1)} = \Pi_{r_1}^{(1)}\Gamma_{r_1}^{(1)^{-1}} \cdots \Pi_{m-1}^{(1)}\Gamma_{m-1}^{(1)^{-1}}U^{(1)} .$$

This is a decomposition of $B^{(1)}$ suitable for our purposes. We could adjust the order of rows in $B^{(1)}$ and multiply out

$$I\Pi_{r_1}^{(1)}\Gamma_{r_1}^{(1)^{-1}} \ldots \Pi_{m-1}^{(1)}\Gamma_{m-1}^{(1)^{-1}}$$

to produce a decomposition

$$\Pi^{(1)}B^{(1)} = L^{(1)}U^{(1)}$$

which copies the decomposition used for B . But this turns out to require more work for the execution of the simplex method cycle than is required if the decomposition is left in the product form given above.

If another execution of the simplex method cycle is taken, the resulting basic matrix $B^{(2)}$ is to be formed from $B^{(1)}$ in the same manner in which $B^{(1)}$ was formed from B . The decomposition for $B^{(2)}$ which we would use in the subsequent cycle would therefore be

$$\Pi B^{(2)} = L[\Pi_{r_1}^{(1)}\Gamma_{r_1}^{(1)^{-1}} \ldots \Pi_{m-1}^{(1)}\Gamma_{m-1}^{(1)^{-1}}] \times$$

$$\times [\Pi_{r_2}^{(2)}\Gamma_{r_2}^{(2)^{-1}} \ldots \Pi_{m-1}^{(2)}\Gamma_{m-1}^{(2)^{-1}}]U^{(2)} .$$

And in general, if $B^{(k)}$ is the k^th basic matrix which we construct, the general decomposition which will concern us will be

$$\Pi B^{(k)} = L[\Pi_{r_1}^{(1)}\Gamma_{r_1}^{(1)^{-1}} \dots \Pi_{m-1}^{(1)}\Gamma_{m-1}^{(1)^{-1}}] \dots [\Pi_{r_k}^{(k)}\Gamma_{r_k}^{(k)^{-1}} \dots \Pi_{m-1}^{(k)}\Gamma_{m-1}^{(k)^{-1}}]U^{(k)} .$$

Each given linear system

$$B^{(k)}v = q$$

is solved by computing the solution to

$$Lt = \Pi q ,$$

carrying out the transformations

$$w = [\Gamma_{m-1}^{(k)}\Pi_{m-1}^{(k)} \dots \Gamma_{r_k}^{(k)}\Pi_{r_k}^{(k)}] \dots [\Gamma_{m-1}^{(1)}\Pi_{m-1}^{(1)} \dots \Gamma_{r_1}^{(1)}\Pi_{r_1}^{(1)}]t ,$$

and solving

$$U^{(k)}v = w .$$

In following chapters we shall show that the round-off errors committed in the above computations can be bounded _a priori_, which was not the case in the standard implementations of the simplex method. And we shall show that the computational effort required for the modified LU implementation compares favorably with the effort required to carry out a representative explicit-inverse implementation.

7. Round-off Error Analysis of the Modified LU Implementation

Wilkinson [12, 13] has analyzed the computational error associated with solving the linear system

$$Bv = q$$

via the LU decomposition of B as follows:

I. It is assumed, without loss of generality, that the rows of B have been prearranged so that the partial pivoting strategy will not alter their ordering.

II The matrices L and U which are computed from B satisfy

$$B + \delta B = LU$$

III. In solving the triangular systems composed from L and U, vectors w and t are produced which satisfy

$$(L+\delta L)w = q \ ,$$
$$(U+\delta U)t = w$$

And t is the computed approximation to v .

IV. Bounds are given for $\|\delta B\|$, $\|\delta L\|$, and $\|\delta U\|$. Thus the vector t has been shown to satisfy

$$(B+\mathcal{E})t = (B+\delta B+L\delta U+\delta LU+\delta L\delta U)t = q ,$$

with a known bound for $\|\mathcal{E}\|$.

We wish to establish a similar result for the decomposition

$$B^{(k)} = L[\Pi_{r_1}^{(1)}\Gamma_{r_1}^{(1)^{-1}} \cdots \Pi_{m-1}^{(1)}\Gamma_{m-1}^{(1)^{-1}}] \cdots [\Pi_{r_k}^{(k)}\Gamma_{r_k}^{(k)^{-1}} \cdots \Pi_{m-1}^{(k)}\Gamma_{m-1}^{(k)^{-1}}]U^{(k)} ,$$

which, for notational convenience, we will also write

$$B^{(k)} = LG^{(k)}U^{(k)} .$$

The following error analysis will be shown to hold:

I. It is assumed that the arrangement of rows in the first basic matrix B is such that no row rearrangement is needed to produce

$$(B+\delta B) = LU$$

computationally by the partial pivoting strategy.

II. At the kth simplex-method cycle the matrices L and $U^{(k)}$ and the product $G^{(k)}$ satisfy

$$B^{(k)} + \delta B^{(k)} = LG^{(k)}U^{(k)} .$$

III. In solving

$$B^{(k)}v = q$$

via the decomposition above, vectors f, w, and t are produced which satisfy

$$(L+\delta L)w = q ,$$
$$(G^{(k)}+\delta G^{(k)})f = w ,$$

and

$$(u^{(k)}+\delta U^{(k)})t = f .$$

And t is the computed approximation to v .

IV. Bounds for the norms of the matrices $\delta B^{(k)}$, δL, $\delta U^{(k)}$, and $\delta G^{(k)}$ are given. Thus the vector t will satisfy

$$(B^{(k)}+\mathcal{E}^{(k)})t = q ,$$

where

$$\begin{aligned}\mathcal{E}^{(k)} = \delta B^{(k)} &+ LG^{(k)}\delta U^{(k)} + L\delta G^{(k)}U^{(k)} \\ &+ \delta LG^{(k)}U^{(k)} + L\delta G^{(k)}\delta U^{(k)} \\ &+ \delta LG^{(k)}\delta U^{(k)} + \delta L\delta G^{(k)}U^{(k)} \\ &+ \delta L\delta G^{(k)}\delta U^{(k)} .\end{aligned}$$

Hence, a bound for $\|\varepsilon^{(k)}\|$ is known.

Chartres and Geuder [2] give the following useful lemma:

If

$$|\zeta_i| < e < 1$$

and

$$0 \le \mu \le \nu \le \sigma ,$$

then

$$\left| \prod_{i=1}^{\mu} (1+\zeta_i) \prod_{i=\mu+1}^{\nu} (1+\zeta_i)^{-1} - 1 \right| < \frac{\nu}{\sigma}\left[\exp\left(\frac{\sigma e}{1-e} \right) - 1\right] .$$

They also implicitly give the corollary:

If

$$\left| \prod_{j=1}^{\varphi} (1+\varphi_j) \prod_{j=\rho+1}^{\kappa} (1+\varphi_j)^{-1} - 1 \right| < M ,$$

and

$$\left| \prod_{i=1}^{\mu} (1+\zeta_i) \prod_{i=\mu+1}^{\nu} (1+\zeta_i)^{-1} - 1 \right| < N ,$$

then

$$\left| \prod_{j=1}^{\rho} (1+\varphi_j) \prod_{j=\rho+1}^{\kappa} (1+\varphi_j)^{-1} \prod_{i=1}^{\mu} (1+\zeta_i) \prod_{i=\mu+1}^{\nu} (1+\zeta_i)^{-1} - 1 \right| < M + N + MN .$$

This is useful if the bounds on the $|\omega_j|$ differ from those on the $|\zeta_i|$.

We proceed to justify the claims made under II, III, and IV just preceding.

Dropping the superscript (k) momentarily for convenience, we review the Chartres-Geuder demonstration that

$$(U+\delta U)t = f ,$$

where t is the approximation to $U^{-1}f$ computed by back-substitution.

The components of t are produced in order from t_m down to t_1 by the computation

$$t_i = fl[(f_i - \sum_{j=i+1}^{m} u_{i,j}t_j)/u_{i,i}] ,$$

with partial sums accumulated, as usual, in extended precision. That is,

$$p_1 = f_i$$

$$p_2 = f_i(1+\eta_1) - u_{i,i+1}t_{i+1}(1+\mu_1)(1+\sigma_1)$$

$$\vdots \qquad \vdots$$

$$p_{m-i+1} = f_i \prod_{\nu=1}^{m-i} (1+\eta_\nu) - \sum_{\ell=i+1}^{m} u_{i,\ell}t_\ell(1+\mu_{\ell-i})(1+\sigma_{\ell-i}) \prod_{\nu=\ell-i+1}^{m-i} (1+\eta_\nu) ,$$

where $|\eta_\nu|$, $|\mu_{\ell-i}|$, and $|\sigma_{\ell-i}|$ are all bounded by ϵ_2 . Finally,

$$t_i = \frac{p_{m-i+1}}{u_{i,i}} (1+\varphi)(1+\rho) ,$$

where φ and ρ are the errors due to the reduction of p_{n-i+1} to normal precision and to the division by $u_{i,i}$, respectively. Thus,

$$f_i = u_{i,i}t_i(1+\varphi)^{-1}(1+\rho)^{-1} \prod_{\nu=1}^{m-i} (1+\eta_\nu)^{-1}$$

$$+ \sum_{\ell=i+1}^{m} u_{i,\ell}t_\ell(1+\mu_{\ell-i})(1+\sigma_{\ell-i}) \prod_{\mu=1}^{\ell-i} (1+\eta_\mu)^{-1} ,$$

or

$$f_i = \sum_{\ell=i}^{m} u_{i,\ell}t_\ell + \sum_{\ell=i}^{m} \delta u_{i,\ell}t_\ell ,$$

where

$$\delta u_{i,i} = [(1+\varphi)^{-1}(1+\rho)^{-1} \prod_{\nu=1}^{m-i} (1+\eta_\nu)^{-1}-1]u_{i,i} ,$$

and

$$\delta u_{i,\ell} = [(1+\mu_{\ell-i})(1+\sigma_{\ell-i}) \prod_{\nu=1}^{\ell-i} (1+\eta_\nu)^{-1}-1]u_{i,\ell}$$

for $\ell > i$. Appealing to the lemma and corollary just given, we may set the following bounds:

$$|\delta u_{i,i}| < \{(m-i)\epsilon_2' + \frac{2\epsilon_1-\epsilon_1^2}{(1-\epsilon_1)^2}[1+(m-i)\epsilon_2']\}|u_{i,i}|$$

and

$$|\delta u_{i,j}| < \{(j-i+2)\epsilon_2'\}|u_{i,j}| \quad \text{for } j > i \ ,$$

where

$$\epsilon_2' = \frac{1}{m+2}\{\exp\left(\frac{(m+2)\epsilon_2}{1-\epsilon_2}\right)-1\} \ .$$

But using the inequality

$$\|D\| \leq m \max_{i,j} |d_{ij}| \ ,$$

which holds for all matrices D (Wilkinson [12]), we may write

$$\|\delta U\| < m\{(m+1)\epsilon_2' + \frac{2\epsilon_1-\epsilon_1^2}{(1-\epsilon_1)^2}[1+(m-1)\epsilon_2']\} \max_{i \leq j} |u_{i,j}| \ .$$

In the same manner it is shown that

$$(L+\delta L)w = q \ ,$$

where

$$|\delta\ell_{ii}| < (i-1)\epsilon_2' + \frac{\epsilon_1}{1-\epsilon_1}[1+(i-1)\epsilon_2'] \ ,$$

and

$$|\delta\ell_{ij}| < |\ell_{ij}|(j+2)\epsilon_2' \quad \text{for} \quad i > j \ .$$

So

$$\|\delta L\| < \{(m+2)\epsilon_2' + \frac{\epsilon_1}{1-\epsilon_1}[1+(m-1)\epsilon_2']\}m \ ,$$

since $\|\delta L\| \leq m \max_{i\leq j} |\delta\ell_{ij}|$, and $|\ell_{ij}| \leq 1$ for all $i \leq j$.

The vector f used above was obtained by the computation

$$f = f\ell\{[\Gamma_{m-1}^{(k)}\Pi_{m-1}^{(k)} \cdots \Gamma_{r_k}^{(k)}\Pi_{r_k}^{(k)}] \cdots [\Gamma_{m-1}^{(1)}\Pi_{m-1}^{(1)} \cdots \Gamma_{r_1}^{(1)}\Pi_{r_1}^{(1)}]w\} \ .$$

This is effected a pair of transformations at a time:

$$\begin{aligned} w^{(0)} &= w \\ w^{(1)} &= f\ell[\Gamma_{r_1}^{(1)}\Pi_{r_1}^{(1)}w^{(0)}] \\ w^{(2)} &= f\ell[\Gamma_{r_1+1}^{(1)}\Pi_{r_1+1}^{(1)}w^{(1)}] \\ &\vdots \quad\quad \vdots \end{aligned}$$

$$\vdots \qquad\qquad \vdots$$

$$w^{(\omega)} = f\ell[\Gamma_{m-1}^{(k)}\Pi_{m-1}^{(k)}w^{(\omega-1)}]$$

$$f = w^{(\omega)} ,$$

where

$$\omega = \sum_{i=1}^{k} (m-r_i) .$$

Thus we wish to make an analysis of

$$d = f\ell(\Gamma\Pi a)$$

for any vector a, where

$$\Gamma = \begin{array}{c|c|c|c|c|}
 & & \nu & \nu+1 & \\
\hline
 & \begin{matrix}1 & & \\ & \ddots & \\ & & 1\end{matrix} & & & \\
\hline
\nu & & 1 & & \\
\hline
\nu+1 & & -g & 1 & \\
\hline
 & & & & \begin{matrix}1 & & \\ & \ddots & \\ & & 1\end{matrix} \\
\hline
\end{array} ,$$

and where Π is either the identity matrix or the identity with the νth and $\nu+1$st rows interchanged.

If Π is the identity, then

$$d = fl(\Gamma\Pi a)$$

is equivalent, compenentwise, to the following:

$$\begin{cases} d_i = a_i \quad \text{for} \quad i \neq \nu + 1 \\ d_{\nu+1} = [a_{\nu+1} - ga_\nu] + [a_{\nu+1}\sigma - ga_\nu(\rho+\tau+\rho\tau)] \ , \end{cases}$$

where $|\sigma|$, $|\rho|$, and $|\tau|$ are all bounded by ε_1. Thus, if we let

$$\varphi = -g(\rho+\tau+\rho\tau) \ ,$$

we have in matrix form $d = (\Gamma+\delta\Gamma)a$, where

$$\delta\Gamma = \begin{array}{|c|c|c|c|l} \multicolumn{1}{c}{} & \multicolumn{1}{c}{\nu} & \multicolumn{1}{c}{\nu+1} & \multicolumn{1}{c}{} & \\ \hline \quad & \quad & \quad & \quad & \\ \hline & & & & \nu \\ \hline & \varphi & \sigma & & \nu+1 \\ \hline & & & & \\ \hline \end{array} \ ,$$

or trivially, since $a = \Pi a$,

$$d = (\Gamma+\delta\Gamma)\Pi a \ .$$

Similarly, if Π is the permuted identity, we find that

$$d = \left\{ \begin{array}{c|c|c|c|c} & \begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix} & & & \\ \hline \nu & & 0 & 1 & \\ \hline \nu+1 & & 1 & -g & \\ \hline & & & & \begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix} \end{array} \right. +$$

$$\left. + \begin{array}{c|c|c|c|c} & \nu & \nu+1 & & \\ \hline & & & & \\ \hline & & & & \nu \\ \hline & \sigma & \varphi & & \nu+1 \\ \hline & & & & \end{array} \right\} a \ ,$$

or

$$d = (\Gamma\Pi+\delta\Gamma\Pi)a \ .$$

So again

$$d = (\Gamma+\delta\Gamma)\Pi a \ .$$

Thus, for

$$f = f\ell\{\Gamma_{m-1}^{(k)}\Pi_{m-1}^{(k)} \ldots \Gamma_{r_1}^{(1)}\Pi_{r_1}^{(1)}w\}$$

we may write

$$f = [\Gamma_{m-1}^{(k)}+\delta\Gamma_{m-1}^{(k)}]\Pi_{m-1}^{(k)} \ldots [\Gamma_{r_1}^{(1)}+\delta\Gamma_{r_1}^{(1)}]\Pi_{r_1}^{(1)}w \ ,$$

or

$$w = \Pi_{r_1}^{(1)}[\Gamma_{r_1}^{(1)}+\delta\Gamma_{r_1}^{(1)}]^{-1} \ldots \Pi_{m-1}^{(k)}[\Gamma_{m-1}^{(k)}+\delta\Gamma_{m-1}^{(k)}]^{-1}f \ .$$

But

$$(\Gamma+\delta\Gamma)^{-1} = \left[\begin{array}{|c|c|c|}\hline I & & \\ \hline & \begin{matrix} 1 & 0 \\ -g+\varphi & 1+\varphi \end{matrix} & \\ \hline & & I \\ \hline\end{array}\right]^{-1}$$

$$= \left[\begin{array}{|c|c|c|}\hline I & & \\ \hline & \begin{matrix} 1 & 0 \\ \frac{g-\varphi}{1+\sigma} & \frac{1}{1+\sigma} \end{matrix} & \\ \hline & & I \\ \hline\end{array}\right]$$

$$= (\Gamma^{-1}+\delta\Gamma^{-1}) ,$$

where

$$\Gamma^{-1} = \left[\begin{array}{c|c|c} I & & \\ \hline & \begin{matrix} 1 & 0 \\ g & 1 \end{matrix} & \\ \hline & & I \end{array}\right]$$

and

$$\delta\Gamma^{-1} = \left[\begin{array}{c|c|c} & & \\ \hline & \begin{matrix} 0 & 0 \\ \frac{-g\sigma+\varphi}{1+\sigma} & \frac{-\sigma}{1+\sigma} \end{matrix} & \\ \hline & & \end{array}\right]$$

Therefore

$$w = [G^{(k)}+\delta G^{(k)}]f \; ,$$

where

$$G^{(k)} = \Pi_{r_1}^{(1)}\Gamma_{r_1}^{(1)^{-1}} \cdots \Pi_{m-1}^{(k)}\Gamma_{m-1}^{(k)^{-1}} ,$$

and

$$\delta G^{(k)} = \Pi_{r_1}^{(1)}[\Gamma_{r_1}^{(1)^{-1}}+\delta\Gamma_{r_1}^{(1)^{-1}}] \cdots \Pi_{m-1}^{(k)}[\Gamma_{m-1}^{(k)^{-1}}+\delta\Gamma_{m-1}^{(k)^{-1}}]-G^{(k)} .$$

Recall at this point that

$$\Pi_i^{(j)}\Pi_i^{(j)} = I$$

for all possible i and j. If we interpose such products at appropriate places among the factors in $\delta G^{(k)}$, we can arrange to express $\delta G^{(k)}$ in the equivalent form

$$\delta G^{(k)} = P^{(k)}[\Omega_{r_1}^{(1)}+\delta\Omega_{r_1}^{(1)}] \cdots [\Omega_{m-1}^{(k)}+\delta\Omega_{m-1}^{(k)}]-P^{(k)}\Omega_{r_1}^{(1)} \cdots \Omega_{m-1}^{(k)} ,$$

where

$$P^{(k)} = \Pi_{r_1}^{(1)} \cdots \Pi_{m-1}^{(k)} ,$$

and

$$\Omega^{(k)}_{m-1} + \delta\Omega^{(k)}_{m-1} = \Gamma^{(k)^{-1}}_{m-1} + \delta\Gamma^{(k)^{-1}}_{m-1} ,$$

$$\Omega^{(k)}_{m-2} + \delta\Omega^{(k)}_{m-2} = \Pi^{(k)}_{m-1}[\Gamma^{(k)^{-1}}_{m-2} + \delta\Gamma^{(k)^{-1}}_{m-2}]\Pi^{(k)}_{m-1} ,$$

$$\begin{array}{ccc} \vdots & & \vdots \end{array}$$

$$\Omega^{(1)}_{r_1} + \delta\Omega^{(1)}_{r_1} = P^{(k)}[\Gamma^{(1)^{-1}}_{r_1} + \delta\Gamma^{(1)^{-1}}_{r_1}]P^{(k)} .$$

This puts the product $\delta G^{(k)}$ into a form to which we can suitably apply the following lemma:

If $H_1, \ldots, H_p$ and $G_1, \ldots, G_p$ are square matrices, then

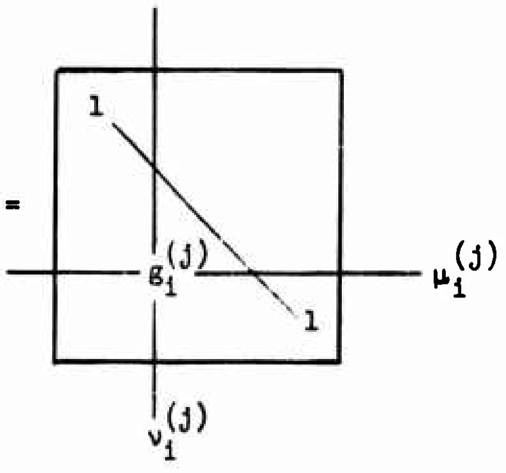
$$\begin{aligned} H_1 \cdots H_p - G_1 \cdots G_p = {} & (H_1 - G_1)G_2 \cdots G_p \\ & + H_1(H_2 - G_2)G_3 \cdots G_p \\ & + \cdots + H_1 \cdots H_{p-1}(H_p - G_p) . \end{aligned}$$

According to this,

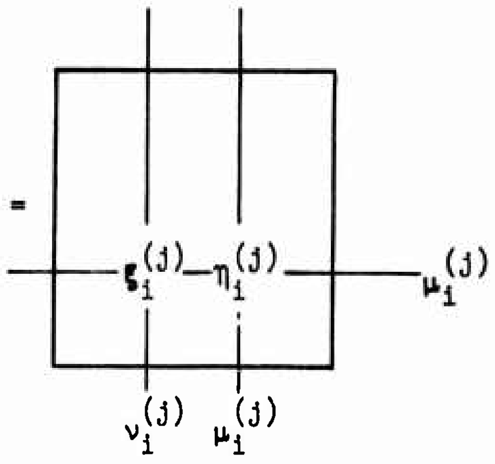
$$\delta G^{(k)} = P^{(k)}\{[\delta\Omega^{(1)}_{r_1}\Omega^{(1)}_{r_1+1} \cdots \Omega^{(k)}_{m-1}] + \cdots$$

$$+ [(\Omega^{(1)}_{r_1}+\delta\Omega^{(1)}_{r_1}) \cdots (\Omega^{(k)}_{m-2}+\delta\Omega^{(k)}_{m-2})\delta\Omega^{(k)}_{m-1}]\} .$$

Each $\Omega^{(j)}_i$ has diagonal elements equal to $+1$ and only one non-zero off-diagonal element, $g^{(j)}_i$, whose magnitude is bounded by one. This element appears somewhere in the subdiagonal portion of the matrix:

$\Omega^{(j)}_i =$  .

The corresponding $\delta\Omega^{(j)}_i$ has only two nonzero entries:

$\delta\Omega^{(j)}_i =$  ,

where

$$\xi_i^{(j)} = -\frac{g_i^{(j)}\sigma_i^{(j)}+\varphi_i^{(j)}}{1+\sigma_i^{(j)}} ,$$

and

$$\eta_i^{(j)} = -\frac{\sigma_i^{(j)}}{1+\sigma_i^{(j)}} .$$

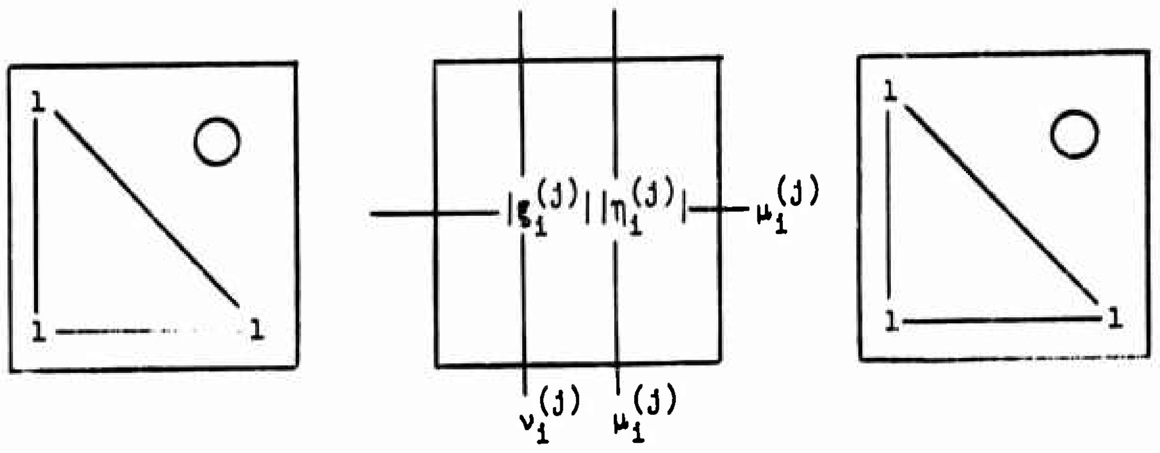
We recall that $|\sigma_i^{(j)}| < \varepsilon_1$ and $|\varphi_i^{(j)}| = |g_i^{(j)}(\rho_i^{(j)}+\tau_i^{(j)}+\rho_i^{(j)}\tau_i^{(j)})|$, where $|\rho_i^{(j)}| < \varepsilon_1$ and $|\tau_i^{(j)}| < \varepsilon_1$. So

$$|\varphi_i^{(j)}| < 2\varepsilon_1 + \varepsilon_1^2 ,$$

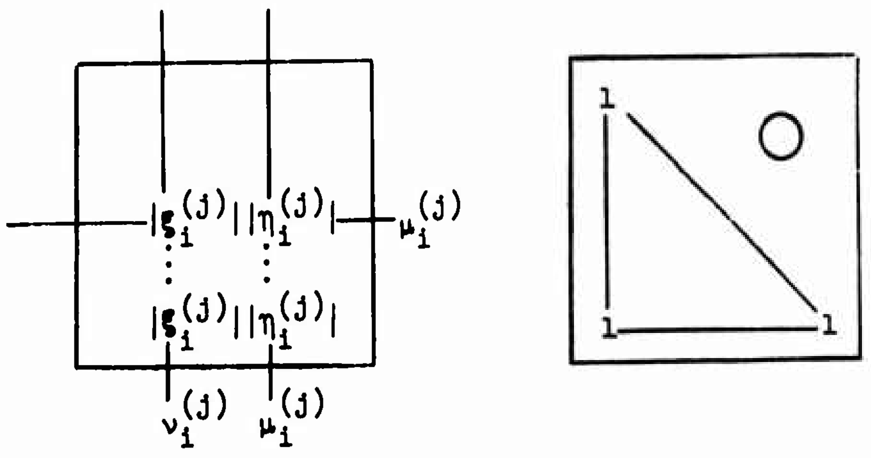
and thus

$$|\xi_i^{(j)}| + |\eta_i^{(j)}| < \frac{4\varepsilon_1+\varepsilon_1^2}{1-\varepsilon_1} .$$

Each term in the sum composing $\delta G^{(k)}$ is bounded elementwise in absolute value by

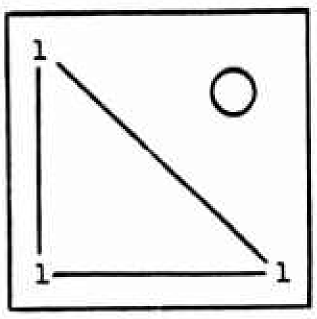


which equals



,

which, in turn, is bounded by

$(|\xi_i^{(j)}|+|\eta_i^{(j)}|)$ 

.

Hence

$$\|\delta G^{(k)}\| \leq \| |\delta G^{(k)}| \| \leq \|P^{(k)}\| \, \| \sum_i \sum_j (|\xi_i^{(j)}| + |\eta_i^{(j)}|) \boxed{\begin{smallmatrix} 1 & & 0 \\ & \ddots & \\ 1 & \cdots & 1 \end{smallmatrix}} \| \, .$$

But

$$\|P^{(k)}\| = \|\Pi_{r_1}^{(1)} \cdots \Pi_{m-1}^{(k)}\| = 1 \, ,$$

and

$$\| \boxed{\begin{smallmatrix} 1 & & 0 \\ & \ddots & \\ 1 & \cdots & 1 \end{smallmatrix}} \| \leq m \, ,$$

and there are

$$\sum_{i=1}^{k} (m-r_i)$$

terms in the expression for $\delta G^{(k)}$. Therefore

$$\|\delta G^{(k)}\| \leq m \frac{4\epsilon_1 + \epsilon_1^2}{1-\epsilon_1} \sum_{i=1}^{k} (m-r_i) \, .$$

Finally we wish to establish the relationship between the basic matrix $B^{(k)}$ and the factors L, $G^{(k)}$, and $U^{(k)}$ of its numerically computed decomposition. We proceed inductively.

If $B^{(0)}$ is the basic matrix initially decomposed, then matrices L and $U^{(0)}$ are computed such that

$$B^{(0)} + \delta B^{(0)} = LU^{(0)} .$$

For the moment we neglect the superscript (0) . The decomposition is produced by letting

$$u_{ij} = fl(b_{ij} - \sum_{\nu=1}^{i-1} \ell_{i\nu} u_{\nu j}) \quad \text{for} \quad i \leq j$$

and

$$\ell_{ij} = fl[(b_{ij} - \sum_{\nu=1}^{j-1} \ell_{i\nu} u_{\nu j})/u_{jj}] \quad \text{for} \quad i > j ,$$

where j is assigned in order the integers from 1 to m, and for each value of j, i is assigned in order the integers from 1 to m . The inner products

$$\sum_{\nu=1}^{i-1} \ell_{i\nu} u_{\nu j} \quad \text{and} \quad \sum_{\nu=1}^{j-1} \ell_{i\nu} u_{\nu j}$$

are, as usual, to be computed in extended-precision arithmetic. Thus, for each pair of values for i and j, if we let

$$\mu = \min\{i,j\} ,$$

then we set

$$p_1 = b_{ij} ,$$

and

$$p_{\nu+1} = p_\nu(1+\eta_\nu) - \ell_{i\nu} u_{\nu j}(1+\sigma_\nu)(1+\zeta_\nu)$$

$$\text{for } \nu = 1, 2, \ldots, \mu - 1 ,$$

where $|\eta_\nu|$, $|\sigma_\nu|$, and $|\zeta_\nu|$ are all bounded by ε_2. Finally, if $i \leq j$ we set

$$u_{ij} = p_\mu(1+\varphi) ,$$

and if $i > j$ we set

$$\ell_{ij} = (p_\mu/u_{jj})(1+\varphi)(1+\rho) ,$$

where $|\varphi| < \varepsilon_1$ represents the error committed in reducing p_μ to normal-precision accuracy, and $|\rho| < \varepsilon_1$ represents the error committed while performing the division. We may also write, for the sake of symmetry,

$$u_{ij} = p_\mu(1+\varphi)(1+\rho) ,$$

if $\rho = 0$ in this case. Thus, in general,

$$\ell_{i\mu} u_{\mu j} = (1+\varphi)(1+\rho) p_\mu ,$$

where we have used the fact that $\ell_{ii} = 1$. We find also that

$$p_\mu = b_{ij} \prod_{\nu=1}^{\mu=1} (1+\eta_\nu)$$

$$- \sum_{\kappa=1}^{\mu-1} \ell_{i\kappa} u_{\kappa j} (1+\sigma_\kappa)(1+\zeta_\kappa) \prod_{\nu=\kappa+1}^{\mu-1} (1+\eta_\nu) \quad .$$

Therefore

$$b_{ij} = \ell_{i\mu} u_{\mu j} (1+\varphi)^{-1} (1+\rho)^{-1} \prod_{\nu=1}^{\mu-1} (1+\eta_\nu)^{-1}$$

$$+ \sum_{\kappa=1}^{\mu-1} \ell_{i\kappa} u_{\kappa j} (1+\sigma_\kappa)(1+\zeta_\kappa) \prod_{\nu=1}^{\kappa} (1+\eta_\nu)^{-1}$$

That is,

$$b_{ij} = \sum_{\kappa=1}^{\mu} \ell_{i\kappa} u_{\kappa j} - \delta b_{ij} ,$$

where

$$\delta b_{ij} = -\ell_{i\mu} u_{\mu j}[(1+\varphi)^{-1}(1+\rho)^{-1} \prod_{\nu=1}^{\mu-1} (1+\eta_\nu)^{-1}-1]$$

$$-\sum_{\kappa=1}^{\mu-1} \ell_{i\kappa} u_{\kappa j}[(1+\sigma_\kappa)(1+\zeta_\kappa) \prod_{\nu=1}^{\kappa} (1+\eta_\nu)^{-1}-1] .$$

Thus

$$|\delta b_{ij}| \leq |\ell_{i\mu} u_{\mu j}|\{(\mu-1)\varepsilon_2' + \frac{2\varepsilon_1+\varepsilon_1^2}{(1-\varepsilon_1)^2} [1+(\mu-1)\varepsilon_2']\}$$

$$+\sum_{\kappa=1}^{\mu-1} |\ell_{i\kappa} u_{\kappa j}|(\kappa+2)\varepsilon_2'$$

And, since

$$\|\delta B\| \leq m \max_{i,j} |\delta b_{ij}| ,$$

we have

$$\|\delta B\| < m^2 \max_{i,j} |u_{ij}|\{(m+1)\varepsilon_2' + \frac{2\varepsilon_1+\varepsilon_1^2}{(1-\varepsilon_1)^2} [1+(m+1)\varepsilon_2']\} .$$

Now suppose that, after k exchanges, we have

$$B^{(k)} + \delta B^{(k)} = LG^{(k)}U^{(k)} .$$

That is, columnwise

$$[B_1^{(k)},\ldots, B_m^{(k)}] + [\delta B_1^{(k)},\ldots, \delta B_m^{(k)}]$$
$$= LG^{(k)}[U_1^{(k)},\ldots, U_m^{(k)}] .$$

Let $B^{(k+1)}$ be formed as described in the preceding chapter. That is, drop column $B_{r_k}^{(k)}$ from $B^{(k)}$, shift the r_k+1st through the mth column left one place, and append the column A_{s_k}. The decomposition is made to reflect this change by modifying $U^{(k)}$. The column $U_{r_k}^{(k)}$ is dropped, the subsequent columns are shifted one position to the left, and the column $H_m^{(k+1)}$, which is the computed approximation to

$$G^{(k)^{-1}}L^{-1}A_{s_k} ,$$

is appended.

From previous discussions we know that $H_m^{(k+1)}$ satisfies

$$(L+\Delta^{(k)})(G^{(k)}+\Xi^{(k)})H_m^{(k+1)} = A_{s_k} ,$$

where

$$\|\Delta^{(k)}\| \leq \{(m+2)\epsilon_2' + \frac{\epsilon_1}{1-\epsilon_1}[1+(m-1)\epsilon_2']\}m ,$$

and

$$\|\Xi^{(k)}\| \leq m \frac{4\epsilon_1+\epsilon_1^2}{1-\epsilon_1} \sum_{i=1}^{k} (m-r_i) .$$

Therefore

$$LG^{(k)}H_m^{(k+1)} = A_{s_k} - [L\Xi^{(k)}+\Delta^{(k)}G^{(k)}+\Delta^{(k)}\Xi^{(k)}]H_m^{(k+1)}$$

$$= A_{s_k} + \delta A_{s_k}$$

So that columnwise

$$[B_1^{(k)},\ldots, B_{r_k-1}^{(k)}, B_{r_k+1}^{(k)},\ldots, B_m^{(k)}, A_{s_k}]$$

$$+ [\delta B_1^{(k)},\ldots, \delta B_{r_k-1}^{(k)}, \delta B_{r_k+1}^{(k)},\ldots, \delta B_m^{(k)}, \delta A_{s_k}]$$

$$= B^{(k+1)} + \Theta^{(k+1)}$$

$$= LG^{(k)}[U_1^{(k)},\ldots, U_{r_k-1}^{(k)}, U_{r_k+1}^{(k)},\ldots, U_m^{(k)}, H_m^{(k+1)}]$$

$$= LG^{(k)}H^{(k+1)} .$$

At this point the transformations

$$\Gamma_{m-1}^{(k+1)}\Pi_{m-1}^{(k+1)} \ldots \Gamma_{r_k}^{(k+1)}\Pi_{r_k}^{(k+1)}$$

are applied to $H^{(k+1)}$ to produce $U^{(k+1)}$. The product of transformations

$$G^{(k)}\Pi_{r_k}^{(k+1)}\Gamma_{r_k}^{(k+1)^{-1}} \cdots \Pi_{m-1}^{(k+1)}\Gamma_{m-1}^{(k+1)^{-1}}$$

is taken as $G^{(k+1)}$.

Columnwise we have

$$\begin{aligned} U_1^{(k+1)} &= H_1^{(k+1)} \\ \vdots \quad & \quad \vdots \\ U_{r_k-1}^{(k+1)} &= H_{r_k-1}^{(k+1)} \end{aligned}$$

and

$$\begin{aligned} U_{r_k}^{(k+1)} &= f\ell\{[\Gamma_{r_k}^{(k+1)}\Pi_{r_k}^{(k+1)}]H_{r_k}^{(k+1)}\} \\ \vdots \quad & \quad \vdots \\ U_{m-1}^{(k+1)} &= f\ell\{[\Gamma_{m-1}^{(k+1)}\Pi_{m-1}^{(k+1)} \cdots \Gamma_{r_k}^{(k+1)}\Pi_{r_k}^{(k+1)}]H_{m-1}^{(k+1)}\} \\ U_m^{(k+1)} &= f\ell\{[\Gamma_{m-1}^{(k+1)}\Pi_{m-1}^{(k+1)} \cdots \Gamma_{r_k}^{(k+1)}\Pi_{r_k}^{(k+1)}]H_m^{(k+1)}\} . \end{aligned}$$

Hence, from previous discussions,

$$\begin{aligned} H_1^{(k+1)} &= U_1^{(k+1)} \\ \vdots \quad & \quad \vdots \\ H_{r_k-1}^{(k+1)} &= U_{r_k-1}^{(k+1)} \end{aligned}$$

and

$$H_{r_k}^{(k+1)} = [\Phi_{r_k}^{(k+1)} + \delta\Phi_{r_k}^{(k+1)}]U_{r_k}^{(k+1)}$$

$$\vdots \qquad\qquad \vdots$$

$$H_m^{(k+1)} = [\Phi_m^{(k+1)} + \delta\Phi_m^{(k+1)}]U_m^{(k+1)} \quad ,$$

where

$$\|\delta\Phi_j^{(k+1)}\| \leq m \frac{4\varepsilon_1 + \varepsilon_1^2}{1-\varepsilon_1}(j+1-r_k) \quad ,$$

with

$$\Phi_j^{(k+1)} = \Gamma_j^{(k+1)}\Pi_j^{(k+1)} \cdots \Gamma_{r_k}^{(k+1)}\Pi_{r_k}^{(k+1)}$$

for $j \neq m$, and

$$\Phi_m^{(k+1)} = \Phi_{m-1}^{(k+1)} \quad .$$

However, because of the special forms which the vectors $U_j^{(k+1)}$ and the matrix $\Phi_m^{(k+1)}$ have, we may replace

$$H_j^{(k+1)} = [\Phi_j^{(k+1)} + \delta\Phi_j^{(k+1)}]U_j^{(k+1)}$$

for each $j = 1, \ldots, m$ by

$$H_j^{(k+1)} = \Phi_m^{(k+1)} U_j^{(k+1)} + \Psi^{(k+1)} ,$$

where

$$\Psi_1^{(k+1)} = \ldots = \Psi_{r_k-1}^{(k+1)} = 0$$

and

$$\Psi_\nu^{(k+1)} = \delta\Phi_\nu^{(k+1)} U_\nu^{(k+1)} \quad \text{for} \quad \nu = r_k, \ldots, m .$$

Thus

$$H^{(k+1)} = \Phi_m^{(k+1)} U^{(k+1)} + \Psi^{(k+1)} ,$$

where $\Psi^{(k+1)}$ is the matrix with columns $\Psi_j^{(k+1)}$. Therefore

$$\begin{aligned} B^{(k+1)} + \Theta^{(k+1)} &= LG^{(k)} H^{(k+1)} \\ &= LG^{(k)} [\Phi_m^{(k+1)} U^{(k+1)} + \Psi^{(k+1)}] . \end{aligned}$$

Or, if $G^{(k+1)} = G^{(k)} \Phi_m^{(k+1)}$,

$$B^{(k+1)} + \Theta^{(k+1)} - LG^{(k)} \Psi^{(k+1)} = LG^{(k+1)} U^{(k+1)} .$$

That is,

$$B^{(k+1)} + \delta B^{(k+1)} = LG^{(k+1)}U^{(k+1)} ,$$

and the induction is complete.

To bound $\|\delta B^{(k+1)}\|$, we note that

$$\Theta^{(k+1)} = [\delta B_1^{(k)}, \ldots, \delta B_{r_k-1}^{(k)}, \delta B_{r_k+1}^{(k)}, \ldots, \delta B_m^{(k)}, \delta A_{s_k}] ,$$

$$\delta A_{s_k} = - [L\Xi^{(k)} + \Delta^{(k)}G^{(k)} + \Delta^{(k)}\Xi^{(k)}]H_m^{(k+1)} ,$$

and

$$\begin{aligned} \Psi^{(k+1)} &= [\Psi_1^{(k+1)}, \ldots, \Psi_{r_k-1}^{(k+1)}, \Psi_{r_k}^{(k+1)}, \ldots, \Psi_m^{(k+1)}] \\ &= [0, \ldots, 0, \delta\Phi_{r_k}^{(k+1)}U_{r_k}^{(k+1)}, \ldots, \delta\Phi_m^{(k+1)}U_m^{(k+1)}] . \end{aligned}$$

Now

$$\begin{aligned} \|\delta A_{s_k}\| &\le (\|L\|\ \|\Xi^{(k)}\| + \|\Delta^{(k)}\|\ \|G^{(k)}\| + \|\Delta^{(k)}\|\ \|\Xi^{(k)}\|)\|H_m^{(k+1)}\| \\ &\le m^2\Bigg[\frac{4\varepsilon_1+\varepsilon_1^2}{1-\varepsilon_1}\sum_{i=1}^{k}(m-r_i) + \{(m+2)\varepsilon_2' + \frac{\varepsilon_1}{1-\varepsilon_1}[1+(m-1)\varepsilon_2']\} \\ &\qquad \{1+\frac{4\varepsilon_1+\varepsilon_1^2}{1-\varepsilon_1}\sum_{i=1}^{k}(m-r_i)\}\Bigg]\|H_m^{(k+1)}\| , \end{aligned}$$

which we denote by $\rho^{(k+1)}$. So, if

$$\mathcal{B}^{(k)} \geq \max_{i,j} |\delta b_{ij}^{(k)}| ,$$

then

$$\max\{\mathcal{B}^{(k)}, \rho^{(k+1)}\} \geq \max_{i,j} |\theta_{ij}| .$$

Furthermore

$$\begin{aligned}\max_{i,j} |\psi_{ij}^{(k+1)}| &\leq \max_{j} \|\delta\Phi_j^{(k+1)} U_j^{(k+1)}\| \\ &\leq \max_{j} \|\delta\Phi_j^{(k+1)}\| \max_{r} \|U_r^{(k+1)}\| \\ &\leq m^2 \frac{4\varepsilon_1+\varepsilon_1^2}{1-\varepsilon_1} (m+1-r_k) \max_{i,j} |u_{ij}^{(k+1)}| ,\end{aligned}$$

which we denote by $\mathcal{R}^{(k+1)}$ Therefore, we may conclude that

$$\max_{i,j} |\delta b_{ij}^{(k+1)}| \leq \max_{i,j} |\theta_{ij}^{(k+1)}| + m \max_{i,j} |\psi_{ij}^{(k+1)}| ,$$

and that

$$\begin{aligned}\|\delta B^{(k+1)}\| &\leq \| \Theta^{(k+1)}\| + \|LG^{(k)}\| \|\Psi^{(k+1)}\| \\ &\leq m[\max\{\mathcal{B}^{(k)}, \rho^{(k+1)}\} + m\mathcal{R}^{(k+1)}] .\end{aligned}$$

To recapitulate: If we are asked to compute the vector v satisfying

$$B^{(k)}v = q$$

via the modified LU decomposition

$$B^{(k)} + \delta B^{(k)} = LG^{(k)}U^{(k)} ,$$

we actually calculate the vector t, an approximation to v, which satisfies

$$(B^{(k)}+\mathcal{E}^{(k)})t = q ,$$

where

$$\begin{aligned}\mathcal{E}^{(k)} = \delta B^{(k)} &+ LG^{(k)}\delta U^{(k)} + L\delta G^{(k)}U^{(k)} \\ &+ \delta LG^{(k)}U^{(k)} + L\delta G^{(k)}\delta U^{(k)} \\ &+ \delta LG^{(k)}\delta U^{(k)} + \delta L\delta G^{(k)}U^{(k)} \\ &+ \delta L\delta G^{(k)}\delta U^{(k)} .\end{aligned}$$

Now

$$\|\delta L\| \leq m\{(m+2)\varepsilon_2' + \frac{\varepsilon_1}{1-\varepsilon_1}[1+(m-1)\varepsilon_2']\} ,$$

$$\|\delta G^{(k)}\| \leq m \frac{4\epsilon_1+\epsilon_1^2}{1-\epsilon_1} \sum_{i=1}^{k} (m-r_i) ,$$

and

$$\|\delta U^{(k)}\| \leq m\{(m-1)\epsilon_2' + \frac{2\epsilon_1-\epsilon_1^2}{(1-\epsilon_1)^2} [1+(m-1)\epsilon_2']\} \max_{i\leq j} |u_{ij}^{(k)}| .$$

And in order to bound $\|\delta B^{(k)}\|$, we let

$$\mathcal{B}^{(0)} = m \max_{i\leq j} |u_{ij}^{(0)}|\{(m+1)\epsilon_2' + \frac{2\epsilon_1+\epsilon_1^2}{(1-\epsilon_1)^2} [1+(m+1)\epsilon_2']\}$$

and

$$\mathcal{B}^{(k+1)} = \max\{\mathcal{B}^{(k)}, \rho^{(k+1)}\} + m\mathcal{R}^{(k+1)}$$

for $k = 0, 1, 2,\ldots$. This makes

$$\|\delta B^{(k)}\| \leq m\mathcal{B}^{(k)}$$

for all k .

Thus it is possible to compute a bound on $\|\mathcal{E}^{(k)}\|$ at every execution of the simplex-method cycle. This will be the foundation for the <u>a posteriori</u> bounds developed in Chapter 9. Moreover, an <u>a priori</u> bound on $\|\mathcal{E}^{(k)}\|$ for all k, expressed only in terms of the given data, could be given at this point. The bound would be extremely large and of no practical value, but its existence argues a numerical stability for the modified LU implementation of the simplex method which was not present in the explicit-inverse implementation. This *a priori* bound would be grounded upon the following considerations, round-off effects in the computation of $u_{ij}^{(k)}$ and $h_{ij}^{(k)}$ being neglected for the moment:

$$k \leq \frac{n!}{m!(n-m)!} ,$$

$$m - r_i \leq m - 1 \quad \text{for all} \quad i ,$$

$$\max_{i \leq j} |u_{ij}^{(0)}| \leq 2^{m-1} \max_{i,j} |a_{ij}|$$

(Wilkinson [13, p. 97]), and

$$\max_{i \leq j} |u_{ij}^{(k)}| \leq m \max_{i,j} |h_{ij}^{(k)}| \quad \text{for all} \quad k = 1, 2, \ldots$$

(Wilkinson [14, p. 218]), where

$$\max_{i,j} |h_{ij}^{(k)}| \leq \max\{\max_{i,j} |h_{ij}^{(k-1)}|, \|H_m^{(k)}\|\} ,$$

and

$$\|H_m^{(k)}\| = \|(G^{(k)}+\Xi^{(k)})^{-1}(L+\Delta^{(k)})^{-1}A_{s_{k-1}}\|$$

$$\leq m \max_{i,j} |a_{ij}| \quad \text{(approximately).}$$

Finally we must note that one of the systems of equations to be solved in the simplex-method cycle uses $B^{(k)^T}$ as the coefficient matrix. It can be shown that, if t is the solution to

$$B^{(k)^T} v = q$$

computed via the modified LU decomposition for $B^{(k)}$, then t satisfies

$$(B^{(k)^T} + \mathcal{F}^{(k)}) t = q ,$$

where any bound on $\|\mathcal{E}^{(k)}\|$ is also a suitable bound for $\|\mathcal{F}^{(k)}\|$.

8. The Standard LU Implementation of the Simplex Method

The round-off error analysis due to Chartres and Geuder [2] which holds for the conventional method of solving a linear system

$$Bv = q$$

via the LU decomposition of B was covered, in passing, throughout the discussion in the preceding chapter. It is worth a quick review for its own sake, since we wish to describe an implementation of the simplex method in which this error analysis is valid.

The decomposition of B is produced by letting

$$u_{ij} = f\ell(b_{ij} - \sum_{\nu=1}^{i-1} \ell_{i\nu} u_{\nu j}) \quad \text{for} \quad i \leq j ,$$

and

$$\ell_{ij} = f\ell[(b_{ij} - \sum_{\nu=1}^{j-1} \ell_{i\nu} u_{\nu j})/u_{jj}] \quad \text{for} \quad i > j .$$

From this we find that L and U satisfy

$$B + \delta B = LU ,$$

where, letting $\mu = \min\{i,j\}$,

$$|\delta b_{ij}| \le |\ell_{i\mu} u_{\mu j}| \{ (\mu-1)\varepsilon_2' + \frac{2\varepsilon_1+\varepsilon_1^2}{(1-\varepsilon_1)^2} [1+(\mu-1)\varepsilon_2'] \}$$

$$+ \sum_{\kappa=1}^{\mu-1} |\ell_{i\kappa} u_{\kappa j}| (\kappa+2)\varepsilon_2' .$$

A vector w satisfying

$$(L+\delta L)w = q$$

is produced by letting

$$w_i = f\ell(q_i - \sum_{j=1}^{i-1} \ell_{ij} w_j)$$

for $i = 1, 2, \ldots, m,$ where

$$|\delta \ell_{ii}| \le (i-1)\varepsilon_2' + \frac{\varepsilon_1}{1-\varepsilon_1} [1+(i-1)\varepsilon_2'] ,$$

and

$$|\delta \ell_{ij}| \le |\ell_{ij}| (j+2)\varepsilon_2' \quad \text{for} \quad i > j .$$

Finally a vector t, approximating v and satisfying

$$(U+\delta U)t = w ,$$

is produced by letting

$$t_i = fl[(w_i - \sum_{j=i+1}^{w} u_{ij}t_j)/\ell_{ii}]$$

for $i = m, m-1, \ldots, 1$, where

$$|\delta\ell_{ii}| \le \{(m-i)\varepsilon_2' + \frac{2\varepsilon_1+\varepsilon_1^2}{(1-\varepsilon_1)^2}[1+(m-i)\varepsilon_2']\}|\ell_{ii}| ,$$

and

$$|\delta u_{ij}| \le (j-i+2)\varepsilon_2'|u_{ij}| \quad \text{for} \quad j > i .$$

Hence the vector t satisfies

$$(B+\mathcal{E})t = q ,$$

where

$$\mathcal{E} = \delta B + \delta LU + L\delta U + \delta L\delta U ,$$

or

$$e_{ij} = \delta b_{ij} + \sum_{\kappa=1}^{\mu} (\delta\ell_{i\kappa}u_{\kappa j}+\ell_{i\kappa}\delta u_{\kappa j}+\delta\ell_{i\kappa}\delta u_{\kappa j}) .$$

So

$$|e_{ij}| \leq |\ell_{i\mu}u_{\mu j}|\{(\mu-1)\epsilon_2' + \frac{2\epsilon_1+\epsilon_1^2}{(1-\epsilon_1)^2}[1+(\mu-1)\epsilon_2']\}$$

$$+ |\delta\ell_{i\mu}u_{\mu j}| + |\ell_{i\mu}\delta u_{\mu j}| + |\delta\ell_{i\mu}\delta u_{\mu j}|$$

$$+ \sum_{\kappa=1}^{\mu-1} |\ell_{i\kappa}u_{\kappa j}|\{(j+\kappa+6)\epsilon_2'+(\kappa+2)(j-\kappa+2)\epsilon_2'^2\} ,$$

where, if we let

$$\lambda = (m-1)\epsilon_2' + \frac{2\epsilon_1+\epsilon_1^2}{(1-\epsilon_1)^2}[1+(m-1)\epsilon_2'] ,$$

we may set the bounds

$$|\delta\ell_{i\mu}u_{\mu j}| \leq \begin{cases} \lambda|u_{ij}| & \text{for } i < j , \\ \lambda|u_{jj}| & \text{for } i = j , \text{ and} \\ (m+1)|\ell_{ij}u_{jj}|\epsilon_2' & \text{for } i > j ; \end{cases}$$

$$|\ell_{i\mu}\delta u_{\mu j}| \leq \begin{cases} (m+1)|u_{ij}|\epsilon_2' & \text{for } i < j , \\ \lambda|u_{jj}| & \text{for } i = j , \text{ and} \\ \lambda|\ell_{ij}u_{jj}| & \text{for } i > j ; \end{cases}$$

and

$$|\delta\ell_{i\mu}\delta u_{\mu j}| \leq \begin{cases} \lambda(m+1)|u_{ij}|\varepsilon_2' & \text{for } i < j\ , \\ \lambda^2|u_{jj}| & \text{for } i = j\ , \text{ and} \\ \lambda(m+1)|\ell_{ij}\;{}_{jj}|\varepsilon_2' & \text{for } i > j\ . \end{cases}$$

This gives the result

$$|e_{ij}| \leq \sum_{\kappa=1}^{\mu-1} |\ell_{i\kappa}u_{\kappa j}|[(j+\kappa+6)+(\kappa+2)(j-\kappa+2)\varepsilon_2']\varepsilon_2'$$

$$+ \begin{cases} [2\lambda+(m+1)\varepsilon_2'+(m+1)\lambda\varepsilon_2']|u_{ij}| & \text{for } i < j\ , \\ [3\lambda+\lambda^2]|u_{jj}| & \text{for } i = j\ , \text{ and} \\ [2\lambda+(m+1)\varepsilon_2'+(m+1)\lambda\varepsilon_2']|\ell_{ij}u_{jj}| & \text{for } i > j\ . \end{cases}$$

Let us set

$$\tilde{\varepsilon}_2 = \varepsilon_2'(1+2\varepsilon_2')$$

and

$$\tilde{\varepsilon}_1 = \lambda + (m+1)\varepsilon_2' + (m+1)\varepsilon_2'\lambda + \lambda^2\ .$$

As estimates, then, we will have

$$\tilde{\varepsilon}_2 \approx \varepsilon_2' \approx \varepsilon_2$$

and

$$\tilde{\varepsilon}_1 \approx \lambda \approx 2\varepsilon_1 .$$

Using these, we may write

$$|e_{ij}| \leq 2(j+3)\tilde{\varepsilon}_2 \sum_{\kappa=1}^{\mu-1} |\ell_{i\kappa} u_{\kappa j}| + \begin{cases} 2\tilde{\varepsilon}_1 |u_{ij}| & \text{for } i < j , \\ 3\tilde{\varepsilon}_1 |u_{jj}| & \text{for } i = j , \text{ and} \\ 2\tilde{\varepsilon}_1 |\ell_{ij} u_{jj}| & \text{for } i > j . \end{cases}$$

This leads to the bound

$$\|\mathcal{E}\| \leq m \max_{i,j} |e_{ij}| \leq m[3\tilde{\varepsilon}_1 + 2m(m+3)\tilde{\varepsilon}_2] \max_{i \leq j} |u_{ij}| .$$

The argument concerning a priori bounds which was given in the last chapter can be applied here as well. That is, by noting that

$$\max_{i \leq j} |u_{ij}| \leq 2^{m-1} \max_{i,j} |a_{ij}| ,$$

we may place an upper bound on $\|\mathcal{E}\|$ a priori, in a simple fashion, in terms of the given data. For the purposes of the a posteriori bounds to be discussed in Chapter 9, the quantity

$$m[3\tilde{\varepsilon}_1 + 2m(m+3)\tilde{\varepsilon}_2] \max_{i \leq j} |u_{ij}|$$

can be computed within each execution of the simplex-method cycle. This quantity, being only a constant multiple of $\max_{i\leq j} |u_{ij}|$, has better behavior than the computed bound on $\|\mathcal{E}^{(k)}\|$ developed in the preceding chapter. For, initially, the modified LU implementation and the standard LU implementation are the same. That is, $\|\mathcal{E}^{(0)}\|$ has the bound which we have just given above. On subsequent executions of the simplex-method cycle, however, terms depending upon the transformations $G^{(k)}$ enter the bound for $\|\mathcal{E}^{(k)}\|$. These terms bring in the quantities

$$\sum_{i=1}^{k} (m-r_i) \quad \text{and} \quad \mathcal{B}^{(k)} ,$$

which grow roughly in a linear fashion with k and cause the bound for $\|\mathcal{E}^{(k)}\|$ gradually to grow larger than the bound given above for $\|\mathcal{E}\|$.

Finally, we should note as in the preceding chapter that if we solve a linear system

$$B^T v = q$$

via the LU decomposition computed for B, then we actually calculate a vector t which satisfies

$$(B^T + \mathcal{F})t = q ,$$

where any bound on $\|\mathcal{E}\|$ is also a suitable bound for $\|\mathcal{F}\|$.

The error analysis given above can be used in the context of the simplex method as follows.

Let

$$B = [A_{\nu_1}, \ldots, A_{\nu_m}]$$

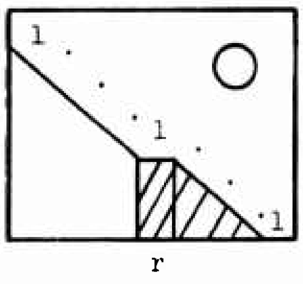
be the current basic matrix, and let L and U be the matrices of its computed decomposition. We assume that the order of the rows in B is such that the partial pivoting strategy produces no row reordering.

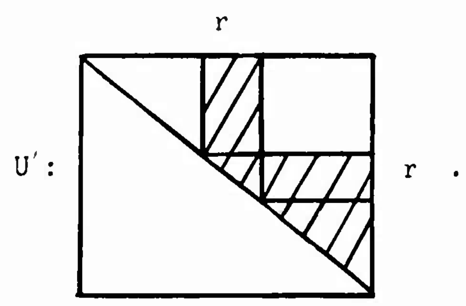
If the simplex method cycle is carried out to produce the new basis

$$\{A_{\nu_1}, \ldots, A_{\nu_{r-1}}, A_{\nu_{r+1}}, \ldots, A_{\nu_m}, A_s\} ,$$

then let the new basis matrix B' be constructed by dropping the r th column of B and introducing A_s in its place. In this case the decomposition matrices L' and U' which would be computed from B' using the partial pivoting strategy differ from L and U only in the shaded areas indicated below:

L′:





Thus, only in the shaded areas must the elements ℓ'_{ij} and u'_{ij} be computed, and only for the last $m - \beta + 1$ rows do row interchanges have to be considered. The saving of work implied by this is considered in detail in Chapter 11. But in rough terms, half the work needed to compute L' and U' from scratch is needed on the average to update the LU decomposition of B to the $L'U'$ decomposition of B'. Even so, this amounts to an order of m^3 operations for the average execution of the simplex method cycle. The conventional implementations of the simplex method as well as the modified LU implementation described previously require only an order of m^2 operations.

9. Error Bounds for Linear-System Solutions Found by LU Decomposition

We have seen that the round-off errors committed in computing the solution v to the linear system

$$Bv = g$$

permit only an approximate solution $v + \delta v$ to be produced which satisfies

$$(B+\mathcal{E})(v+\delta v) = g$$

for some matrix $\mathcal{E}$ whose norm we can bound. Thus

$$\delta v = - B^{-1}\mathcal{E}(v+\delta v) ,$$

and so

$$\|\delta v\| \leq \|B^{-1}\| \; \|\mathcal{E}\| \; \|v+\delta v\| \; .$$

Since $\|v+\delta v\|$ can be computed and an upper bound on $\|\mathcal{E}\|$ is available, we can bound $\|\delta v\|$ by finding an upper bound for

$$\|B^{-1}\| = \sqrt{\frac{1}{\lambda_{\min}(BB^T)}} \; ;$$

i.e., by finding a lower bound for $\lambda_{min}(BB^T)$.

Let the eigenvalues of BB^T be

$$0 < \lambda_1 = \lambda_{min}(BB^T) \le \lambda_2 \le \cdots \le \lambda_m .$$

We can compute the sum of these eigenvalues as

$$d = \sum_{i=1}^{m} \lambda_i = tr(BB^T) = \sum_{i=1}^{m} \sum_{j=1}^{m} b_{ij}^2 .$$

Further, if we have solved the given linear system via some decomposition

$$B + \delta B = VW ,$$

where $\det(V)$ and $\det(W)$ are particularly easy to compute, and a bound for $\|\delta B\|$ is available, then we can compute an approximation to the product of the eigenvalues as well:

$$\begin{aligned} p &= [\det(B+\delta B)]^2 \\ &= \prod_{i=1}^{m} \lambda_i([B+\delta B][B+\delta B]^T) \\ &= \prod_{i=1}^{m} \lambda_i(BB^T+\delta BB^T+B\delta B^T+\delta B\delta B^T) . \end{aligned}$$

But BB^T and $\delta BB^T + B\delta B^T + \delta B\delta B^T$ are both symmetric matrices. Hence, it follows that

$$\lambda_i(BB^T+\delta BB^T+B\delta B^T+\delta B\delta B^T) = \lambda_i + \delta\lambda_i$$

for each i, where

$$|\delta\lambda_i| \le \|\delta BB^T+B\delta B^T+\delta B\delta B^T\| .$$

(See Wilkinson [14, Chapter 2, §44].) Suppose, therefore, that

$$K \ge \|\delta BB^T+B\delta B^T+\delta B\delta B^T\| .$$

Then

$$|\delta\lambda_i| \le K$$

for all i . And

$$p = \prod_{i=1}^{m} (\lambda_i+\delta\lambda_i) ,$$

$$d = \sum_{i=1}^{m} (\lambda_i+\delta\lambda_i) - \sum_{i=1}^{m} \delta\lambda_i .$$

The best lower bound for $\lambda_1 = \lambda_{min}(BB^T)$ which we could hope to obtain using this information would be the component q_1 of the solution to the following programming problem:

$$\begin{cases} \text{minimize} & f(q_1, \ldots, q_m) = q_1 \\ \text{subject to} & g_1(q_1, \ldots, q_m) = \sum_{i=1}^{m} q_i - d - mK \leq 0 \\ & g_2(q_1, \ldots, q_m) = \sum_{i=1}^{m} q_i - d + mK \geq 0 \\ & g_3(q_1, \ldots, q_m) = \sum_{i=1}^{m} \log(q_i) - \log(p) = 0 \end{cases}$$

Using the method of Lagrange multipliers, we are presented with four linear systems to solve:

1.

$$\begin{cases} \sum_{i=1}^{m} \log(q_i) - \log(p) = 0 \\ 1 - \mu_1/q_1 = 0 \\ - \mu_1/q_2 = 0 \\ \vdots \qquad \vdots \\ - \mu_1/q_m = 0 \end{cases}$$

which has no solution;

2.

$$\begin{cases} \sum_{i=1}^{m} \log(q_i) - \log(p) = 0 \\ \sum_{i=1}^{m} q_i - d - mK = 0 \\ 1 - \mu_1/q_1 - \mu_2 = 0 \\ - \mu_1/q_2 - \mu_2 = 0 \\ \vdots \qquad \vdots \\ - \mu_1/q_m - \mu_2 = 0 \end{cases} ,$$

from which we first infer that

$$q_2 = \dots = q_m = Q ,$$

giving the equations

$$\begin{cases} q_1 Q^{m-1} = p \\ q_1 + (m-1)Q = d + mK , \end{cases}$$

which imply that q_1 must satisfy

$$q_1 = p\left(\frac{m-1}{d+mK-q_1} \right)^{m-1} ;$$

3.

$$\begin{cases} \sum_{i=1}^{m} \log(q_i) - \log(p) = 0 \\ \sum_{i=1}^{m} q_i - d + mK = 0 \\ 1 - \mu_1/q_1 - \mu_2 = 0 \\ - \mu_1/q_2 - \mu_2 = 0 \\ \qquad \vdots \qquad \vdots \\ - \mu_1/q_m - \mu_2 = 0 , \end{cases}$$

whose solution, as with the preceding system, has a first component which must satisfy

$$q_1 = p\left(\frac{m-1}{d-mK-q_1} \right)^{m-1} ;$$

and

4.
$$\begin{cases} \sum_{i=1}^{m} \log(q_i) - \log(p) = 0 \\ \sum_{i=1}^{m} q_i - d - mK = 0 \\ \sum_{i=1}^{m} q_i - d + mK = 0 \\ 1 - \mu_1/q_1 - \mu_2 - \mu_3 = 0 \\ - \mu_1/q_2 - \mu_2 - \mu_3 = 0 \\ \qquad \vdots \qquad\qquad \vdots \\ - \mu_1/q_m - \mu_2 - \mu_3 = 0 , \end{cases}$$

from which we first infer that

$$q_2 = \cdots = q_m = Q ,$$

giving the equations

$$\begin{cases} q_1 Q^{m-1} = p \\ q_1 + (m-1)Q = d + mK \\ q_1 + (m-1)Q = d - mK , \end{cases}$$

which have no solution.

Thus we use the lower bound

$$\lambda_1 \geq \max\{q_+, q_-\} ,$$

where q_+, q_- satisfy

$$q_{\pm} = p\left(\frac{m-1}{d \pm mK - q_{\pm}}\right)^{m-1} .$$

If F is a real matrix of order m with positive eigenvalues, the fixed point f of the equation

$$f = \det(F)\left(\frac{m-1}{tr(F)-f} \right)^{m-1}$$

is often used as a lower bound for $\lambda_{min}(F)$ (Bodewig [1, page 69]). The fixed point is found iteratively with $f = 0$ inserted initially into the right-hand side. With this background, it is easily shown that

$$q_- = \max\{q_+, q_-\} .$$

For, in the iterative production of q_+ and q_- we let

$$q_+^{(0)} = q_-^{(0)} = 0$$

and

$$q_\pm^{(i+1)} = p\left(\frac{m-1}{d\pm mK-q_\pm^{(i)}}\right)^{m-1}$$

for $i = 0, 1, \ldots$. Thus, we proceed inductively. Initially,

$$q_-^{(0)} = q_+^{(0)} .$$

Suppose

$$q_-^{(i)} \geq q_+^{(i)}$$

for some $i \geq 1$. Then

$$d + mK - q_+^{(i)} \geq d + mK - q_-^{(i)} \geq d - mK - q_-^{(i)} .$$

So

$$p\left(\frac{m-1}{d-mK-q_-^{(i)}}\right)^{m-1} \geq p\left(\frac{m-1}{d+mK-q_+^{(i)}}\right)^{m-1} ,$$

or

$$q_-^{(i+1)} \geq q_+^{(i+1)} .$$

Hence

$$q_- = \lim_{i \to \infty} q_-^{(i)} \geq \lim_{i \to \infty} q_+^{(i)} = q_+ .$$

Therefore, if $Z \geq \|e\|$, we can bound $\|\delta v\|$ by

$$\|\delta v\| \leq \sqrt{1/q_-}\; Z \|v+\delta v\| .$$

10. Employment of Error Bounds in the Simplex-Method Cycle

Each study which we have made on an implementation of the simplex method has led us to conclude that, in solving the systems

$$\begin{cases} B^T\pi = \gamma \\ Bu = b \\ By = A_s \end{cases}$$

computationally, round-off errors permit us to produce only the approximate vectors

$$\pi + \delta\pi,\ u + \delta u,\ y + \delta y\ .$$

From these studies we have also determined how bounds on the errors

$$\begin{cases} \|\delta\pi\| \leq \omega \\ \|\delta u\| \leq \eta \\ \|\delta y\| \leq \varphi \end{cases}$$

can be computed.

Furthermore, in computing the objective function

$$z = \pi^T b\ ,$$

we can produce only

$$\begin{aligned} z + \delta z &= (\pi+\delta\pi)^T b \\ &= \pi^T b + \delta\pi^T b \\ &= z + \delta\pi^T b \ , \end{aligned}$$

so that, letting

$$\xi = \omega \|b\| \ ,$$

$$|\delta z| = |\delta\pi^T b| \leq \|\delta\pi\| \ \|b\| \leq \xi \ .$$

Likewise, in computing

$$\bar{c}_j = \pi^T A_j - c_j$$

for $j \notin \{\nu_1, \dots, \nu_m\}$, $1 \leq j \leq n$, we can only produce $\bar{c}_j + \delta\bar{c}_j$, where

$$|\delta\bar{c}_j| = |\delta\pi^T A_j| \leq \zeta_j$$

for $\zeta_j = \omega\|A_j\|$.

These bounds may be used to give warnings that round-off effects have become harmful. As an example, at each execution of the simplex-method cycle the test that B is, indeed, a basic matrix is that $u > 0$. Conceivably round-off errors committed during the previous execution of the simplex-method cycle can have caused the wrong column to be moved to or from the basis. As a check on u, then, we note that, for any i,

$$u_i > 0 \text{ if } u_i + \delta u_i > \eta ,$$

and

$$u_i < 0 \text{ if } u_i + \delta u_i < -\eta .$$

If neither of the above hold, then we shouldn't draw conclusions about the sign of u_i. Thus, if for any i it is discovered that

$$u_i + \delta u_i \leq \eta ,$$

then the computation in the previous execution of the simplex-method cycle bears a second look.

Similarly an improper exchange of columns in the previous execution of the simplex-method cycle may have produced a basic matrix, but one for which the associated objective function value shows a decrease rather than the increase which it should. Let $\hat{z} + \delta\hat{z}$ and $\hat{\xi}$ be respectively the previously computed objective function value and associated error bound. Let $z + \delta z$ and ξ be the corresponding values for the current execution. Then

$$\hat{z} < z \text{ if } \hat{z} + \delta\hat{z} + \hat{\xi} < z + \delta z + \xi ,$$

and

$$z < \hat{z} \quad \text{if} \quad z + \delta z + \xi < \hat{z} + \delta\hat{z} + \xi .$$

If neither holds, the situation is indeterminate. Thus, if

$$z + \delta z + \xi \leq \hat{z} + \delta\hat{z} + \xi ,$$

the computation of the preceding simplex-method cycle could stand review.

Likewise the index s for which

$$\bar{c}_s = \min\{\bar{c}_j\}$$

has definitely been found if

$$\bar{c}_s + \delta\bar{c}_s + \zeta_s \leq \bar{c}_j + \delta\bar{c}_j - \zeta_j$$

for all $j \notin \{\nu_1, \ldots, \nu_m\}$ and $1 \leq j \leq n$. And $\bar{c}_s$ is definitely negative if

$$\bar{c}_s + \delta\bar{c}_s + \zeta_s < 0 .$$

The execution of the simplex-method cycle can proceed if a value of s is found such that only this last inequality holds. Otherwise each $\bar{c}_j$ is nonnegative or has indeterminate sign, and as good a check as possible should be made on the optimality of the current basic feasible solution.

If there is no value of j for which

$$y_j + \delta y_j > \varphi ,$$

then each y_j is negative or has indeterminate sign, and as good a check as possible should be made on the sign of each component of y to establish whether or not the objective function is unbounded from above. If for each j the sign of y_j can be determined; i.e., either

$$y_j + \delta y_j > \varphi$$

or

$$y_j + \delta y_j < - \varphi ,$$

then, if r satisfies

$$\frac{u_r + \delta u_r + \eta}{y_r + \delta y_r - \varphi} \leq \frac{u_j + \delta u_j - \eta}{y_j + \delta y_j + \varphi}$$

for all j such that $y_j + \delta y_j > \varphi$, r is definitely an index such that

$$\frac{u_r}{y_r} = \min\{ \frac{u_j}{y_j} \mid j \text{ is any index for which } y_j > 0 \} .$$

Thus, tests involving η, ω, and φ can be made throughout execution of the simplex-method cycle to determine whether the computations which have just been performed need to be investigated. The needed investigations must be carried out by producing better approximations to the vectors u, π, and y corresponding either to the current basis or to the preceeding one, and computing smaller error bounds η, ω, and φ in association. This is most easily accomplished by iteratively refining the three vectors (Wilkinson [13, pp. 121-127], Forsythe and Moler [5, Chapters 13 and 22], Moler [7]).

Under the usual circumstances; i.e., if ε_0 is sufficiently smaller than ε_1 and the basic matrix is not too badly conditioned, iterative refinement can be expected to improve the approximations to u, π, and y essentially to full normal-precision accuracy. In other terms, η, ω, and φ can each be reduced to a small integer multiple of ε_1 by the iterative refinement. Since the given data A, b, and c are only assumed good to normal-precision accuracy, there is no reason to produce approximations to u, π, and y having more accuracy than that provided by iterative refinement. If difficulties arise which cannot be cleared up by iterative refinement, it is reasonable to argue that they should remain unresolved pending acquisition of more accurate data.

We summarize these remarks by presenting a variation on the simplex-method cycle:

Main Section

1. Produce $u + \delta u$ and η.

   If $u_i + \delta u_i \leq \eta$ for any i, restore the basis used for the previous execution of the cycle, and go to step 8.

2. Produce $\pi + \delta\pi$ and ω .

3. Produce $z + \delta z$ and ξ .

   Let $\hat{z} + \delta\hat{z}$ and $\hat{\xi}$ be the corresponding values produced during the preceding execution of the cycle. If $z + \delta z + \xi \leq \hat{z} + \delta\hat{z} - \hat{\xi}$, restore the preceding basis, and go to step 8.

4. Produce $\bar{c}_j + \delta\bar{c}_j$ and ζ_j for each $j \notin \{\nu_1, \ldots, \nu_m\}$, $1 \leq j \leq n$. For one such value of j set

$$t = \bar{c}_j + \delta\bar{c}_j - \zeta_j \quad \text{and} \quad s = j \ .$$

   For each other such value of j in turn set

$$t = \bar{c}_j + \delta\bar{c}_j - \zeta_j \quad \text{and} \quad s = j$$

   if

$$\bar{c}_j + \delta\bar{c}_j + \zeta_j < t \ .$$

   Having finished, if

$$\bar{c}_s + \delta\bar{c}_s \geq - \zeta_s \ ,$$

   then go to step 9.

5. Produce $y + \delta y$ and φ .

   If $y_i + \delta y_i \leq \varphi$ for all i, go to step 9.

6. For one value of i such that $y_i + \delta y_i > \varphi$ let

$$t = \frac{u_i+\delta u_i-\eta}{y_i+\delta y_i+\varphi} \quad \text{and} \quad r = i .$$

For each other such value of i in turn set t and r as above whenever

$$\frac{u_i+\delta u_i+\eta}{y_i+\delta y_i-\varphi} < t .$$

7. Drop A_{ν_r} from the basis and add A_s. Then form a corresponding basic matrix, update a decomposition, or do whatever else is necessary to reflect the basis change, and go to step 1.

Refinement Section

8. Form the necessary basic matrix, compute a decomposition, and produce $u + \delta u$, $\pi + \delta\pi$, $y + \delta y$.

9. Refine $u + \delta u$ and produce a corresponding error bound η. If $u_j + \delta u_j \leq \eta$ for any j, computation should be suspended, and an indication should be made that the problem seems to be degenerate.

10. Refine $\pi + \delta\pi$ and produce a corresponding error bound ω.

11. For each $j \notin \{\nu_1, \ldots, \nu_m\}$, $1 \leq j \leq u$ produce $\bar{c}_j + \delta\bar{c}_j$ and ζ_j from the refined $\pi + \delta\pi$ and ω. Find s as in step 4. If $\bar{c}_s + \delta\bar{c}_s \geq \zeta_s$, the optimal basis has been found.

If $\zeta_s \geq \bar{c}_s + \delta\bar{c}_s \geq -\zeta_s$, the current basis is probably, though not necessarily, optimal.

12. Refine $y + \delta y$ and produce a corresponding error bound φ. If $y_i + \delta y_i \leq \varphi$ for all i, computation should be suspended, and an indication should be made that the objective function appears to be unbounded from above. Otherwise go to step 6.

We should note in closing that, if an implementation of the simplex method constructed along the lines given above produces a vector $\tilde{x} + \delta\tilde{x}$ as the computed approximation to an optimal solution, there is no way of checking its closeness to any true optimum for the problem using the information at hand. This is due to the fact that the components of optimal solutions to linear programming problems need not vary continuously with perturbations in the data. And round-off errors committed during computation of the basic feasible solutions may be expressed, we have seen, as perturbations of the basic matrices, and hence of the data. A bound on $\delta\tilde{x}$ is useful only for delimiting the distance from some basic feasible solution $\tilde{x}$ and the computed vector $\tilde{x} + \delta\tilde{x}$. The problem can quite possibly have a unique optimum x^* for which the distance

$$\|x^* - (\tilde{x} + \delta\tilde{x})\|$$

is greater than the bound on $\|\delta\tilde{x}\|$. Such a situation might arise, for example, in a problem having one optimal point and one or more widely separated nearly optimal points. Then round-off errors can quite possibly cause the computed approximation to one of the nearly optimal points to have a higher associated value of the objective function than does the approximation to the optimal point.

11. Comparison of Implementations by Operation Counts

It is our final goal to compare the computational effort required by each of the three simplex-method implementations which have been studied. These implementations differ only in the manner in which each produces the vectors u, π, and y. Hence, a measure of the work done by each to produce these three vectors will constitute a satisfactory comparison.

Traditionally the number of multiplications and divisions (product operations) needed to carry out a computation has been the measure of the work needed for that computation. The reason for this lies in the much slower speed at which earlier generations of computers have performed product operations in contrast to their speed of performing additions and subtractions (summation operations). As machines are built with more nearly uniform speeds for all arithmetic operations, this tradition is becoming indefensible. In the programs discussed here, however, computations are composed of inner products or of additions of a multiple of one vector to another. Thus, each product operation generally gets paired with a summation operation, and the convenience of counting only the product operations may be kept.

We begin by considering the standard LU implementation, since several of the operation counts which we derive for it are applicable to the modified LU implementation as well.

Consider solving a lower triangular system

$$Lv = g$$

if the first $k - 1$ components of v are known. The components $v_k, \ldots, v_m$ are produced by setting

$$v_i = g_i - \sum_{j=1}^{i-1} \ell_{ij} v_j$$

in order for $i = k, \ldots, m$. This requires

$$(k-1) + (k) + \ldots + (m-1) = \frac{m(m-1)-(k-1)(k-2)}{2}$$

product operations.

Consider solving a lower triangular system

$$U^T v = g$$

if the first $k - 1$ components of v are known. Here, for $i = k, \ldots, m$ in order we set

$$v_i = (g_i - \sum_{j=1}^{i-1} u_{ij} v_j)/u_{ii} ,$$

so that the cost to produce v is

$$k + (k+1) + \ldots + m = \frac{m(k+1)-k(k-1)}{2}$$

product operations.

To produce the vector u in the standard LU implementation, we solve, in order,

$$Lt = b ,$$

and

$$Uu = t .$$

But the matrix L is identical in its first $r_0 - 1$ columns with the lower triangular factor of the basic matrix used in the previous execution of the simplex-method cycle. Thus, if the vector t is saved from the previous execution of the cycle, since the vector b is constant throughout, only the components $t_{r_0}, \ldots, t_m$ need be produced anew, which costs

$$\frac{m(m-1)-(r_0-1)(r_0-2)}{2}$$

product operations.

In solving $Uu = t$, no reduction in effort is afforded by having an old copy of u available. The cost to produce u will be equal to the cost of producing all components of v from the system

$$U^T v = g ;$$

that is,

$$\frac{m(m+1)}{2}$$

product operations.

To produce the vector π we solve, in order,

$$U^T w = \gamma ,$$

and

$$L^T \pi = w .$$

But the matrix U^T is identical in its principal submatrix of order $r_0 - 1$ with the corresponding matrix used on the preceding execution of the simplex-method cycle. Similarly, the vector γ does not differ in its first $r_0 - 1$ components from the corresponding vector used on the preceding cycle. Thus, as was done in producing u, an old copy of w can be used to cut the cost of computing π to

$$\frac{m(m+1)-r_0(r_0-1)}{2} + \frac{m(m-1)}{2}$$

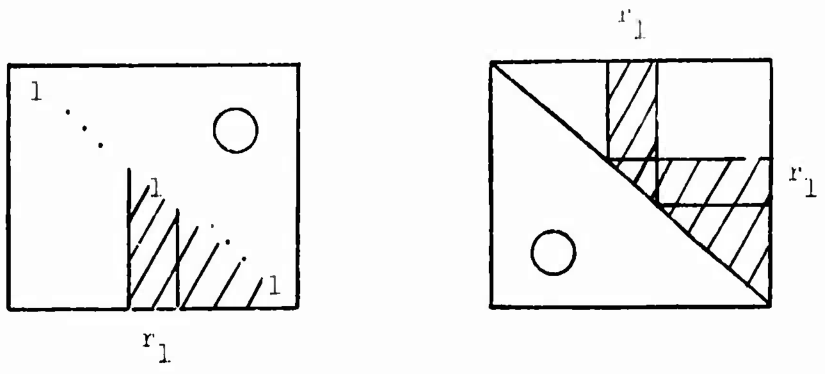
product operations.

No saving in the production of y can be arranged. The total cost required for the computation will be

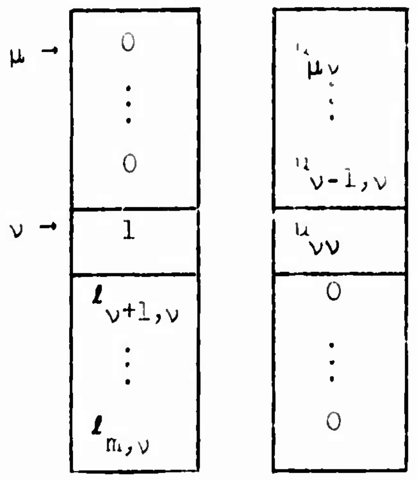
$$\frac{m(m-1)}{2} + \frac{m(m+1)}{2}$$

product operations.

Finally, if the execution of the simplex-method cycle results in the change of column r_1 of the basic matrix, we must count the cost of updating the LU decomposition. The areas which change are shown shaded next:



Now, a partial column of the LU decomposition,



,

is computed by first letting $i = \mu, \mu + 1, \ldots, \nu$ in order and setting

$$u_{i\nu} = b_{i\nu} - \sum_{j=1}^{i-1} \ell_{ij} u_{j\nu} ,$$

which requires

$$(\mu-1) + \mu + \ldots + (\nu-1) = \frac{\nu(\nu-1)-(\mu-1)(\mu-2)}{2}$$

product operations, and then letting $i = \nu + 1, \ldots, m$ in order and setting

$$\ell_{i\nu} = (b_{i\nu} - \sum_{j=1}^{\nu-1} \ell_{ij} u_{j\nu})/u_{\nu\nu} ,$$

which requires

$$(m-\nu)\nu$$

product operations.

To completely update the LU decomposition of the basic matrix, it is required that column r_1 be fully recomputed, and that columns $r_1 + 1$ through m be recomputed from row number r_1 on down. This costs

$$\frac{r_1(r_1-1)}{2} + (m-r_1)r_1 + \sum_{k=r_1+1}^{m} \left[\frac{k(k-1)-(r_1-1)(r_1-2)}{2} + (m-k)k \right]$$

product operations.

Using the equalities

$$\sum_{k=r_1+1}^{m} \frac{k(k-1)}{2} = \frac{1}{6} [(m+1)m(m-1)-(r_1+1)r_1(r_1-1)] ,$$

and

$$\sum_{k=r_1+1}^{m} (m-k)k = m \sum_{k=r_1+1}^{m} k - \sum_{k=r_1+1}^{m} k^2$$

$$= (m-1) \sum_{k=r_1+1}^{m} k - \sum_{k=r_1+1}^{m} k(k-1)$$

$$= \frac{m-1}{2} [m(m+1)-r_1(r_1+1)] - \frac{1}{3} [m(m+1)(m-1)-r_1(r_1+1)(r_1-1)] ,$$

we find that the cost of updating is

$$\frac{r_1(r_1-1)}{2} + r_1(m-r_1) + \frac{1}{6} [(m+1)m(m-1)-(r_1+1)r_1(r_1-1)]$$

$$- \frac{(r_1-2)(r_1-1)(m-r_1)}{2} + \frac{m-1}{2} [m(m+1)-r_1(r_1+1)]$$

$$- \frac{1}{3} [(m+1)m(m-1)-(r_1+1)r_1(r_1-1)]$$

$$= \frac{m^3}{3} - mr_1^2 + \frac{2}{3} r_1^3 + 2mr_1 - \frac{3}{2} r_1^2 - \frac{4}{3} m + \frac{5}{6} r_1$$

product operations. Thus, to fully carry out the simplex-method cycle using the standard LU implementation requires

$$\frac{m^3}{3} - mr_1^2 + \frac{2}{3} r_1^3 + 2mr_1 - \frac{3}{2} r_1^2 - \frac{4}{3} m + \frac{5}{6} r_1$$

$$+ \frac{m(m-1)-(r_0-1)(r_0-2)}{2} + \frac{m(m+1)}{2}$$

$$+ \frac{m(m+1)-r_0(r_0-1)}{2} + \frac{m(m-1)}{2}$$

$$+ \frac{m(m-1)}{2} + \frac{m(m+1)}{2}$$

product operations, which simplifies to

$$\frac{m^3}{3} - mr_1^2 + \frac{2}{3} r_1^3 + 2mr_1 - \frac{3}{2} r_1^2 - \frac{4}{3} m + \frac{5}{6} r_1$$

$$+ 3m^2 - r_0^2 + 2r_0 - 1$$

product operations.

This expression is not as useful as we would like for comparison purposes, depending as it does upon the present and preceding exchange through the numbers r_0 and r_1. However, it is reasonable to assume that each value for r_0 and r_1 between 1 and m is equally likely to occur. Hence, the expected number of product operations in any execution of the simplex-method cycle would be

$$\frac{m^3}{3} + 3m^2 - \frac{4}{3} m - 1 + \frac{1}{m}\left[\frac{2}{3} \sum_{\nu=1}^{m} \nu^3 - (m + \frac{5}{2}) \sum_{\nu=1}^{m} \nu^2 + (2m + \frac{17}{6}) \sum_{\nu=1}^{m} \nu\right]$$

or

$$\frac{m^3}{6} + 3m^2 - \frac{m}{6}$$

product operations.

For the modified LU implementation some savings can be made in the computation of the vectors u and π like the savings made in the standard LU implementation.

To produce the vector u we solve

$$Lt = b ,$$

and then set

$$q^{(k)} = [\Gamma_{m-1}^{(k)}\Pi_{m-1}^{(k)} \dots \Gamma_{r_k}^{(k)}\Pi_{r_k}^{(k)}] \dots [\Gamma_{m-1}^{(1)}\Pi_{m-1}^{(1)} \dots \Gamma_{r_1}^{(1)}\Pi_{r_1}^{(1)}]t ,$$

and lastly solve

$$U^{(k)}u = q^{(k)} .$$

However, on the previous execution of the simplex-method cycle we have already solved

$$Lt = b ,$$

and have set

$$q^{(k-1)} = [\Gamma_{m-1}^{(k-1)}\Pi_{m-1}^{(k-1)} \dots \Gamma_{r_{k-1}}^{(k-1)}\Pi_{r_{k-1}}^{(k-1)}] \dots [\Gamma_{m-1}^{(1)}\Pi_{m-1}^{(1)} \dots \Gamma_{r_1}^{(1)}\Pi_{r_1}^{(1)}]t .$$

Hence, if $q^{(k-1)}$ is saved, we need only set

$$q^{(k)} = [\Gamma_{m-1}^{(k)}\Pi_{m-1}^{(k)} \dots \Gamma_{r_k}^{(k)}\Pi_{r_k}^{(k)}]q^{(k-1)} ,$$

and then solve

$$U^{(k)}u = q^{(k)} .$$

The production of $q^{(k)}$ costs $m - r_k$ product operations, and the production of u costs $m(m+1)/2$.

To compute π, we solve

$$U^{(k)^T} w = \gamma ,$$

then we set

$$p^{(k)} = [\Pi_{r_1}^{(1)} \Gamma_{r_1}^{(1)^T} \cdots \Pi_{m-1}^{(1)} \Gamma_{m-1}^{(1)^T}] \cdots [\Pi_{r_k}^{(k)} \Gamma_{r_k}^{(k)^T} \cdots \Pi_{m-1}^{(k)} \Gamma_{m-1}^{(k)^T}] w$$

and solve

$$L^T \pi = p^{(k)} .$$

Noting, as with the standard LU decomposition, what things remain unchanged from one cycle to the next, we find that the first step need cost only

$$\frac{m(m+1) - r_k(r_k - 1)}{2}$$

product operations. The second step costs

$$\sum_{i=1}^{k} (m-r_i) = km - \sum_{i=1}^{k} r_i$$

product operations, and the third step costs

$$\frac{m(m-1)}{2}$$

product operations.

No saving in the production of y can be arranged, so that vector's cost will be

$$\frac{m(m-1)}{2} + \frac{m(m+1)}{2} + km - \sum_{i=1}^{k} r_i$$

product operations.

Suppose the current execution of the simplex-method cycle ends in the dropping of column number r_{k+1} from the basis matrix. Then, in order to update the basis matrix decomposition, we must construct the transformations

$$\Gamma_{m-1}^{(k+1)}\Pi_{m-1}^{(k+1)} \cdots \Gamma_{r_{k+1}}^{(k+1)}\Pi_{r_{k+1}}^{(k+1)}$$

and apply them to the upper Hessenberg matrix $H^{(k+1)}$ to produce $U^{(k+1)}$, as was described in chapter 6. Letting $\sigma = r_{k+1}, \ldots, m-1$ in order, we would carry out the updating by:

1. interchanging rows σ and $\sigma + 1$ if necessary,
2. making one division to compute the Gauss multiplier $g_{\sigma}^{(k+1)}$,
3. subtracting $g_{\sigma}^{(k+1)}$ times the σ^{th} row of $H^{(k+1)}$ from the $\sigma + 1^{th}$ row.

In making this last step, only components $\sigma + 1$ through m of the $\sigma + 1^{st}$ row need to be computed. Thus, for each value of σ we require $m + 1 - \sigma$ product operations for a total of

$$(m+1-r_{k+1}) + \ldots + (m+1-m+1) = 2 + 3 + \ldots + (m+1-r_{k+1})$$

$$= \frac{(m-r_{k+1})(m-r_{k+1}+3)}{2} \quad .$$

Thus, the measure of the work needed to carry out the k^{th} execution of the simplex-method cycle using the modified LU implementation is

$$\frac{m^2}{2} - mr_{k+1} + \frac{r_{k+1}^2}{2} + \frac{3m}{2} - \frac{3r_{k+1}}{2}$$

$$+ \frac{m^2}{2} + \frac{m}{2} + m - r_k$$

$$+ \frac{m^2}{2} + \frac{m}{2} - \frac{r_k^2}{2} + \frac{r_k}{2} + km - \sum_{i=1}^{k} r_i + \frac{m^2}{2} - \frac{m}{2}$$

$$+ \frac{m^2}{2} - \frac{m}{2} + km - \sum_{i=1}^{k} r_i + \frac{m^2}{2} + \frac{m}{2}$$

product operations, which condenses to

$$3m^2 - mr_{k+1} + \frac{r_{k+1}^2}{2} - \frac{r_k^2}{2} + (2k+3)m - \frac{3r_{k+1}}{2}$$

$$- \frac{r_k}{2} - 2 \sum_{i=1}^{k} r_i \ .$$

Again assuming that each value of r_j $(j=1,\ldots, k+1)$ between 1 and m is equally likely, we would expect to perform

$$3m^2 + (2k+3)m + \frac{1}{m}(m-2-2k) \sum_{\nu=1}^{m} \nu$$

$$= \frac{7m^2}{2} + \frac{5m}{2} + km - k - 1$$

product operations on an average execution of the simplex-method cycle using the modified LU decomposition.

For the standard, explicit-inverse implementation of the simplex-method cycle described in chapter 3, the order of proceeding is slightly different. The cycle begins with u and π already at hand, and the vector y is produced by forming the matrix-vector product

$$B^{-1}A_s = y \ ,$$

which costs m^2 product operations.

After the exchange of column A_s for A_{ν_r} in the basis, columns 1 through m of the matrix D must be updated by applying to each the Gauss-Jordan elimination which reduces the vector

$$t = \begin{bmatrix} \bar{c}_s \\ \hline y \end{bmatrix} = \begin{bmatrix} t_0 \\ \hline t_1 \\ \vdots \\ t_m \end{bmatrix}$$

to the r th unit vector. That is, the r th row of D is divided by t_r ; then, for $i = 0,\ldots, r-1, r+1,\ldots, m$, t_i times the r th row is subtracted from the i th row of D. In all, the updating requires $m(m+1)$ product operations.

Finally, the same Gauss-Jordan elimination is applied to the vector

$$\begin{bmatrix} z \\ \hline u \end{bmatrix}$$

at a cost of $m + 1$ product operations.

Thus, one execution of the simplex-method cycle using this explicit-inverse implementation costs

$$2m^2 + 2m + 1$$

product operations. This is to be compared with the

$$\frac{7m^2}{2} + \frac{5m}{2} + km - k - 1$$

product operations needed on the average to perform the k th execution of the simplex-method cycle using the modified LU implementation. And it is to be compared with the

$$\frac{m^3}{6} + 3m^2 - \frac{m}{6}$$

product operations needed on the average to carry out the simplex-method cycle using the standard LU implementation.

Since k does not generally grow larger than a small multiple of m, the modified LU implementation does not require significantly more work than the classical implementation. Guaranteed numerical stability, in short, needn't be expensive.

Bibliography

[1] Bodewig, E.: Matrix calculus, Interscience Publishers, 1956.

[2] Chartres, Bruce A. and Geuder, James C.: Computable error bounds for direct solution to linear equations, J. Assoc. Comp. Mach., vol. 14, Jan. 1967, pp. 63-71.

[3] Clasen, R. J.: Techniques for automatic tolerance control in linear programming, Comm. Assoc. Comp. Mach., vol. 9, no. 11, Nov. 1966, pp. 802-803.

[4] Dantzig, George B.: Linear programming and extensions, Princeton University Press, 1965.

[5] Forsythe, George and Moler, Cleve B.: Computer solution of linear algebraic systems, Prentice-Hall, 1967.

[6] Hadley, G.: Linear programming, Addison Wesley, 1962.

[7] Moler, Cleve B.: Iterative refinement in floating point, J. Assoc. Comp. Mach., vol. 14, Apr. 1967, pp. 316-321.

[8] Mueller-Merbach, Heiner: On round-off errors in linear programming, paper presented at the joint CORS-ORSA conference, May 28, 1964, Montreal.

[9] Rabinowitz, Philip: Applications of linear programming to numerical analysis, SIAM J. Numer. Anal., to appear.

[10] Storøy, Sverre: Error control in the simplex-technique, BIT, vol. 7, 1967, pp. 216-225.

[11] Wilkinson, J. H.: Rounding errors in algebraic processes, Proceedings of the International Conference on Information Processing, UNESCO, 1959.

[12] Wilkinson, J. H.: Error analysis of direct methods of matrix inversion, J. Assoc. Comp. Mach., vol. 8, 1961, pp. 281-330.

[13] Wilkinson, J. H.: Rounding errors in algebraic processes, Prentice-Hall, 1963.

[14] Wilkinson, J. H.: The Algebraic eigenvalue problem, Oxford University Press, 1965.

[15] Wolfe. Philip: Error in the solution of linear programming problems, in Error in digital computation, Louis B. Rall ed., Wiley, 1965.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Computer Science Department Stanford University Stanford, California 94305	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP ---

3. REPORT TITLE

A NUMERICAL INVESTIGATION OF THE SIMPLEX METHOD

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Manuscript for Publication (Technical Report)

5. AUTHOR(S) *(Last name, first name, initial)*

Bartels, Richard H.

6. REPORT DATE July 31, 1968	7a. TOTAL NO. OF PAGES 122	7b. NO. OF REFS 15

8a. CONTRACT OR GRANT NO. N00014-67-A-0112-0029	9a. ORIGINATOR'S REPORT NUMBER(S) CS 104
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* none
d.	

10. AVAILABILITY/LIMITATION NOTICES

Releasable without limitations on dissemination.

11. SUPPLEMENTARY NOTES ---	12. SPONSORING MILITARY ACTIVITY Office of Naval Research

13. ABSTRACT

This report analyzes the behavior of the round-off errors associated with three different computer implementations of the simplex method of linear programming. One of the three is representative of computer implementations in common use, and it is shown that the standard method of updating the basic-matrix inverse is numerically unstable. The remaining two implementations are suggested by the author and use triangular decompositions of the basic matrix as substitutes for its inverse. The implementations are shown to be stable, and one of them is shown to be competitive in speed with the standard simplex-method computer implementations. Error bounds which may be calculated from intermediate results are developed for each of the three implementations, and their use during computation for error monitoring and control is discussed.

DD FORM 1 JAN 64 1473

Unclassified

Security Classification